

Va

```
PPPPPPPP      AAAAAA      FFFFFFFFFF      PPPPPPP      CCCCCCCC      AAAAAA      LL      LL
PPPPPPPP      AAAAAA      FFFFFFFFFF      PPPPPPP      CCCCCCCC      AAAAAA      LL      LL
PP      PP      AA      AA      FF      PP      PP      CC      AA      AA      LL      LL
PP      PP      AA      AA      FF      PP      PP      CC      AA      AA      LL      LL
PP      PP      AA      AA      FF      PP      PP      CC      AA      AA      LL      LL
PP      PP      AA      AA      FF      PP      PP      CC      AA      AA      LL      LL
PPPPPPPP      AA      AA      FFFFFFFFFF      PPPPPPP      CC      AA      AA      LL      LL
PPPPPPPP      AA      AA      FFFFFFFFFF      PPPPPPP      CC      AA      AA      LL      LL
PP      AAAAAAAAAA      FF      PP      CC      AAAAAAAAAA      LL      LL
PP      AAAAAAAAAA      FF      PP      CC      AAAAAAAAAA      LL      LL
PP      AA      AA      FF      PP      CC      AA      AA      LL      LL
PP      AA      AA      FF      PP      CC      AA      AA      LL      LL
PP      AA      AA      FF      PP      CC      AA      AA      LL      LL
PP      AA      AA      FF      PP      CC      AA      AA      LL      LL
PP      AA      AA      FF      PP      CCCCCCCC      AA      AA      LLLLLLLLLL      LLLLLLLLLL
PP      AA      AA      FF      PP      CCCCCCCC      AA      AA      LLLLLLLLLL      LLLLLLLLLL
```

```
LL      IIIIII      SSSSSSSS
LL      IIIIII      SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLL      IIIIII      SSSSSSSS
```


(2)	173	DEFINITIONS
(3)	204	UNIMPLEMENTED FORK PROCESS CALLS
(4)	211	CONNECTION MANAGEMENT CALLS
(4)	212	- FPC\$CONNECT, COMPLETE PROCESSING A CONNECT
(5)	302	- FPC\$ACCEPT, COMPLETE PROCESSING AN ACCEPT
(6)	400	- FPC\$REJECT, PROCESS A REJECT CALL
(7)	441	- FPC\$DISCONNECT, PROCESS A DISCONNECT CALL
(8)	608	SEQUENCED MESSAGE CALLS
(8)	609	- FPC\$ALLOCMSG, ALLOCATE A MESSAGE BUFFER
(9)	667	- FPC\$RCHMSGBUF, RECYCLE MESSAGE BUFFER
(9)	668	- AT HIGH PRIORITY
(9)	669	- FPC\$RCLMSGBUF, RECYCLE MESSAGE BUFFER
(9)	670	- AT LOW PRIORITY
(10)	733	- FPC\$DEALLOCMSG, DEALLOCATE A MESSAGE BUFFER
(10)	734	- FPC\$DEALRGMSG, DEALLOCATE A MESSAGE BUFFER,
(10)	735	- ARGUMENTS PASSED IN REGISTERS
(11)	818	- FPC\$SENDMSG, SEND A SEQUENCED MESSAGE
(12)	903	DATAGRAM SERVICE CALLS
(12)	904	- FPC\$ALLOCDG, ALLOCATE A DATAGRAM BUFFER
(13)	945	- FPC\$DEALLOCDG, DEALLOCATE A DATAGRAM BUFFER
(13)	946	- TO NONPAGED POOL
(14)	975	- FPC\$QUEUEDG, QUEUE A SYSAP SUPPLIED BUFFER
(14)	976	- TO THE DATAGRAM FREE QUEUE
(15)	1005	- FPC\$QUEUEMDGS, ALLOCATE DG'S AND QUEUE FOR
(15)	1006	- RECEIVES OR
(15)	1007	- DEQUEUE DG'S AND RETURN TO
(15)	1008	- NONPAGED POOL
(16)	1099	- FPC\$SENDDG, SEND DATAGRAM
(16)	1100	- FPC\$SENDRGDG, SEND DG, NO CDRP
(17)	1184	BLOCK TRANSFER CALLS
(17)	1185	- FPC\$MAP, MAP A BUFFER
(17)	1186	- FPC\$MAPBYPASS, MAP A BUFFER W/
(17)	1187	- NO ACCESS CHECKING
(17)	1188	- FPC\$MAPIRP, MAP A BUFFER W/
(17)	1189	- ARGUMENTS IN IRP
(17)	1190	- FPC\$MAPIRPBYP, MAP A BUFFER W/
(17)	1191	- ARGUMENTS IN IRP AND NO
(17)	1192	- ACCESS CHECKING
(18)	1333	- FPC\$REQDATA, BLOCK XFER READ
(18)	1334	- FPC\$SENDDATA, BLOCK XFER WRITE
(19)	1475	- UNMAP, UNMAP A BUFFER
(20)	1546	- SUSP_CONCALL, SUSPEND CONNECTION
(20)	1547	- MANAGEMENT CALL
(21)	1580	- STATE_ERR, RETURN CDT STATE ERROR
(21)	1581	- TO SYSAP
(22)	1596	MAINTENANCE FUNCTION CALLS
(22)	1597	- FPC\$READCOUNT, READ AND LOCK
(22)	1598	- PORT COUNTERS
(23)	1687	- FPC\$RLSCOUNT, READ AND RELEASE
(23)	1688	- PORT COUNTERS
(24)	1723	- FPC\$MRESET, RESET REMOTE PORT/SYSTEM
(25)	1752	- FPC\$MSTART, SEND START TO REMOTE
(25)	1753	- SYSTEM
(26)	1808	- FPC\$STOP_VCS, SEND SHUTDOWN ON ALL VCS
(27)	1836	RECEIVED PACKET ROUTINES
(27)	1837	- FPC\$REC_DGREC, PROCESS RECEIVED DG
(28)	1882	- FPC\$REC_SNDDG, PROCESS SENT DG
(29)	1923	- FPC\$REC_DATREC, PROCESS RECEIVED RETDAT
(29)	1924	- FPC\$REC_CNFRUC, PROCESS RECEIVED RETCNF
(30)	1994	- FPC\$REC_MSGREC, PROCESS RECEIVED MSG

(31)	2059	-	FPC\$REC_SNDMSG, PROCESS SEND MSG	H 1
(32)	2096	-	FPC\$REC_RDCNT, PROCESS RECEIVED RDCNT	
(33)	2136	MISC. ROUTINES		
(33)	2137	-	FPC\$CHK_SCONID, CHECK SENDER CONID	
(33)	2138	-	FPC\$CHK_DCONID, CHECK DESTINATION CONID	
(33)	2139	-	FPC\$CHK_LCONID, CHECK CONID IN LCONID	
(34)	2225	FPC\$INITIAL,	INITIALIZE AT THIS LAYER	
(34)	2226	-	BUILD BDT	


```
0000 1 .TITLE PAFPCALL.
0000 2 .IDENT 'V04-001'
0000 3
0000 4 *****
0000 5 *
0000 6 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 * ALL RIGHTS RESERVED.
0000 9 *
0000 10 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 * TRANSFERRED.
0000 16 *
0000 17 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 * CORPORATION.
0000 20 *
0000 21 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 *
0000 24 *
0000 25 *****
0000 26
0000 27 ++
0000 28
0000 29 FACILITY:
0000 30
0000 31 VAX/VMS EXECUTIVE, I/O DRIVERS
0000 32
0000 33 ABSTRACT: SCS ROUTINES AVAILABLE TO FORK PROCESSES WHICH
0000 34 ARE CI PORT-SPECIFIC.
0000 35
0000 36 AUTHOR: N. KRONENBERG, MAY 1981
0000 37
0000 38 MODIFIED BY:
0000 39
0000 40 V04-001 NPK3066 N. Kronenberg 9-Sep-1984
0000 41 Upon deallocation of a message buffer that results
0000 42 in the decision to extend more credit, bypass call
0000 43 to SCSSREQ_SCSSSEND to extend credit if the CDT state
0000 44 shows that the SYSAP has done a DISCONNECT. (Formerly
0000 45 the SCSSREQ_SCSSSEND call was bypassed iff the CDT was
0000 46 actually queued for SCS sending already. This is
0000 47 incorrect since it would allow a credit to be extended
0000 48 after the DISCONNECT_REQUEST was sent.)
0000 49 RSPID mismatch on completion of a block transfer
0000 50 (RD_SEQ_ERR) corrected to back msg pointer up by
0000 51 PPD-header length prior to crashing port.
0000 52
0000 53 V03-025 NPK3054 N. Kronenberg 24-Jun-1984
0000 54 Since SCSSREQ_SCSSSEND will now ensure that a CDT
0000 55 will not be queued on the SCS send buffer wait queue
0000 56 if it is already waiting, change DISCONNECT from the
0000 57 open state not to check for this condition. The
```


0000 58 : check is being moved to SCSSREQ SCSSSEND because there
0000 59 : were several other conditions that required the check
0000 60 : that were not making it and that could corrupt the
0000 61 : wait queue.
0000 62 :
0000 63 : V03-024 NPK3047 N. Kronenberg 22-Mar-1984
0000 64 : Add FPC\$STOP_VCS entry to send host shutdowns to
0000 65 : to all vcs on shutdown or bugcheck.
0000 66 :
0000 67 : V03-023 NPK3048 N. Kronenberg 16-Mar-1984
0000 68 : Fix FPC\$SND CNTMSG to set retflag=true by putting
0000 69 : 1 in R0 instead of SYSAP\$C_DISPPD.
0000 70 :
0000 71 : V03-022 NPK3046 N. Kronenberg 7-Mar-1984
0000 72 : Improve comments for FPC\$READCOUNT.
0000 73 :
0000 74 : V03-021 TMK0002 Todd M. Katz 21-Feb-1984
0000 75 : Change FPC\$INITIAL so that the buffer descriptors are allocated
0000 76 : by calling EX\$ALONONPAGED instead of IN\$HIPALC. This can be
0000 77 : done because this routine is now being called at fork IPL
0000 78 : instead of at IPL\$_POWER.
0000 79 :
0000 80 : V03-020 TMK0001 Todd M. Katz 29-Jan-1984
0000 81 : Fix an error path for the MRESET and MSTART fork process
0000 82 : calls. In both cases when the appropriate PPD action routine
0000 83 : returns an error, the error path that is taken does a PUSHHR of
0000 84 : R0 (instead of a PUSHHL) to save the return status over the
0000 85 : datagram buffer deallocation. This PUSHHR results in the stack
0000 86 : being corrupted in a variety of interesting fashions depending
0000 87 : upon the error code that is residing in R0.
0000 88 :
0000 89 : V03-019 NPK3039 N. Kronenberg 11-Jan-1984
0000 90 : On receipt of DATREC, CNFREC return the response msg
0000 91 : to pool unconditionally. Previously it was returned
0000 92 : to the msg free queue if that queue was not up to
0000 93 : the initial receive credit and this could cause credits
0000 94 : to build without bound.
0000 95 : Fix RD_SEQ_ERR and SC_SEQ_ERR to first look up the
0000 96 : PB (if any) associated with the response in hand, and
0000 97 : then branch to INT\$/RSP_CRASH_PORT which expects R1
0000 98 : to have the PB address or 0 if no PB.
0000 99 :
0000 100 : V03-018 NPK3037 N. Kronenberg 11-Nov-1983
0000 101 : Add \$DEBUGCHECK on block xfer XCTID sequence number
0000 102 : error and source conid sequence number error.
0000 103 : Fix source connection id check to not delete a sent
0000 104 : message twice.
0000 105 :
0000 106 : V03-017 NPK3036 N. Kronenberg 21-Oct-1983
0000 107 : Correct bug in stack management in FPC\$MSTART.
0000 108 :
0000 109 : V03-016 NPK3034 N. Kronenberg 13-Sep-1983
0000 110 : Fix stepping count of number of bytes mapped to add
0000 111 : from byte count pointed to by R1 rather than IRP.
0000 112 :
0000 113 : V03-015 NPK3029 N. Kronenberg 14-Jul-1983
0000 114 : Enhancements for V4.0.


```
0000 115 : Set local/remote process names in scs msg attached to
0000 116 : CDT when connect is issued rather than waiting for accept.
0000 117 : Add per connection performance counters.
0000 118 : Correct benign bug in msg deallocation in deciding
0000 119 : whether to return buffer to pool or free queue.
0000 120 : In FPC$SNDCNTMSG with no rspid decide if port should
0000 121 : put sent buffer on free queue before sending it.
0000 122 : Add new entry FPC$SNDRGDG to send a dg without a CDRP.
0000 123 : Remove NPK3026 since it is taken care of by zeroing
0000 124 : CDRP$L_MSG_BUF at the time the block xfer is started.
0000 125 :
0000 126 : V03-014 NPK3026 N. Kronenberg 18-May-1983
0000 127 : Fix FPC$REC_CNFREC/DATREC to zero CDRP$L_MSG_BUF.
0000 128 :
0000 129 : NPK3025 N. Kronenberg 18-May-1983
0000 130 : Fix the fix to insufficient memory on ACCEPT call.
0000 131 :
0000 132 : V03-013 KTA3046 Kerbey T. Altmann 28-Mar-1983
0000 133 : Redo for SCS/PPD split.
0000 134 :
0000 135 : V03-012 NPK3017 N. Kronenberg 28-Feb-1983
0000 136 : Fix R0 destroyed on READ counters busy.
0000 137 :
0000 138 : V03-011 NPK3016 N. Kronenberg 28-Feb-1983
0000 139 : Fix insufficient dg/msg buffers on ACCEPT call.
0000 140 :
0000 141 : V03-010 NPK3010 N. Kronenberg 11-Nov-1982
0000 142 : Invoke $SYSAPDEF. Add dg disposal flag value assumes.
0000 143 : Fix insfmem path in FPC$MSTART.
0000 144 :
0000 145 : V03-009 NPK3009 N. Kronenberg 2-Nov-1982
0000 146 : Zero application dg credit field.
0000 147 :
0000 148 : V03-008 NPK3008 N. Kronenberg 6-Oct-1982
0000 149 : Change disconnect on CDT in illegal state to crash
0000 150 : the VC instead of returning error status to caller
0000 151 : and doing nothing. Change disconnect on CDT in
0000 152 : disc ack state to crash VC instead of simple unilateral
0000 153 : break of connection.
0000 154 :
0000 155 : V03-007 NPK3007 N. Kronenberg 5-Oct-1982
0000 156 : Fixed bug in MAP which incorrectly saved the context
0000 157 : of multiple buffer descriptor waiters.
0000 158 :
0000 159 : V03-006 NPK3006 N. Kronenberg 9-Sep-1982
0000 160 : Fixed bug in waiting for buffer descriptor.
0000 161 :
0000 162 : V03-005 KDM0002 Kathleen D. Morse 28-Jun-1982
0000 163 : Added $DYNDEF, $DCDEF, $PRDEF, and $SSDEF.
0000 164 :
0000 165 : V03-004 NPK3002 N. Kronenberg 1-Jul-1982
0000 166 : Fix ACCEPT to return correct status in R0 on
0000 167 : insufficient memory and to preserve addr of listen
0000 168 : CDT.
0000 169 :
0000 170 :
0000 171 :--
```


DEFINITIONS

```
0000 173      .SBTTL  DEFINITIONS
0000 174
0000 175      :
0000 176      : Set PSECT to driver code:
0000 177      :
0000 178
00000000 179      .PSECT  $$$115_DRIVER, LONG
0000 180
0000 181      :
0000 182      : System definitions (LIB.MLB):
0000 183      :
0000 184
0000 185      .nocross
0000 186      $CDLDEF      : Connection descriptor list
0000 187      $CDRPDEF   : Class driver request packet format
0000 188      $CDTDEF    : Connection descriptor format
0000 189      $CIBDDEF    : CI buffer descriptor format
0000 190      $CIBDTDEF  : CI buffer desc table format
0000 191      $CIBHANDEF  : CI buffer handle format
0000 192      $DYNDEF    : Dynamic block codes
0000 193      $IRPDEF    : Define IRP offsets and bits
0000 194      $PBDEF     : Path Block format
0000 195      $PDTDEF    : Port descriptor format
0000 196      $PRDEF     : Define processor register definitions
0000 197      $RDDEF     : Response descriptor format
0000 198      $RDTDEF    : Response descriptor list
0000 199      $SCSDEF    : SCS message format
0000 200      $SSDEF     : System-wide status codes
0000 201      $SYSAPDEF  : Send/recvdg flags
0000 202      .cross
```


UNIMPLEMENTED FORK PROCESS CALLS

.SBTTL UNIMPLEMENTED FORK PROCESS CALLS

		0000	204		
		0000	205		
		0000	206	FPC\$MAINTFCN::	
		0000	207		
50	00F4 8F	3C	0000	208	MOVZWL #SS\$_ILLIOFUNC,R0 ; Set error status for caller
		05	0005	209	RSB ; Return to caller

CONNECTION MANAGEMENT CALLS

```
0006 211 .SBTTL CONNECTION MANAGEMENT CALLS
0006 212 .SBTTL - FPC$CONNECT, COMPLETE PROCESSING A CONNECT
0006 213
0006 214 :+
0006 215 : This routine is JMP'ed to from SCSS$CONNECT with a CDT allocated
0006 216 : (and in the closed state) and initialized with the SYSAP's
0006 217 : connect parameters or 0's for fields not yet used. FPC$CONNECT
0006 218 : does port-specific processing: allocates SCS control message
0006 219 : receive buffer, initial credit worth of receive message buffers,
0006 220 : and initial datagram buffers. FPC$CONNECT then sets the CDT
0006 221 : state to connect sent and queues the CDT to send a CONNECT REQ
0006 222 : message to the remote system. Finally, FPC$CONNECT suspends the
0006 223 : SYSAP.
0006 224
0006 225 : Inputs:
0006 226
0006 227 : R3 -Addr of CDT
0006 228 : R4 -Addr of PDT
0006 229
0006 230 : CDT initialized as follows:
0006 231
0006 232 : CDT$L_LCONID -Local conid
0006 233 : MSGINPUT -Addr to call in SYSAP for rec'd msgs
0006 234 : DGINPUT -Addr to call in SYSAP for rec'd dgs
0006 235 : ERRADDR -Addr to call in SYSAP for connection errors
0006 236 : RSTATION -Remote station addr
0006 237 : PDT -Addr of PDT
0006 238 : MINSEND -Minimum send credit req'd by SYSAP
0006 239 : INITLREC -Initial credit extended by SYSAP
0006 240 : DGREC -Initial # of dg's queued
0006 241 : STATE -CLOSED
0006 242 : PB -Addr of selected PB to remote system
0006 243 : WAITQFL/BL -Set to show no entries
0006 244 : RPROCNAM -Addr of dest process name
0006 245 : LPROCNAM -Addr of local process name
0006 246 : CONDAT -Addr of connect data
0006 247
0006 248 : other CDT fields -0
0006 249
0006 250
0006 251 : (SP) -Return PC in SYSAP
0006 252
0006 253 : Outputs:
0006 254
0006 255 : R0 -Status: SSS_NORMAL, SSS_FAILRSP,
0006 256 : SSS_REJECT, SSS_INSFMEM
0006 257 : R1 -Reject reason or fail response reason
0006 258 : if R0 = REJECT or FAILRSP
0006 259 : R2 -Addr of ACCEPT_REQ if R0 = success
0006 260 : other registers -Preserved
0006 261 :-
0006 262
0006 263 : .ENABL LSB
0006 264
0006 265 FPC$CONNECT::
0006 266
0006 267 $CHK_CDTSTATE - ; Verify that CDT state
```



```
0006 268
0006 269
FFEE' 30 000F 270 BSBW
52 32 50 E9 0012 271 BLBC
50 2C A3 D0 0015 272 MOVL
50 54 A3 D0 0019 273 CDT$L_SCSMSG(R3),R2
04 A2 80 7D 001D 274 MOVL
0C A2 80 7D 0021 275 CDT$L_LPROCNAME(R3),R0
50 50 A3 D0 0025 276 MOVQ
14 A2 80 7D 0029 277 (R0)+,SCS$T-DST-PROC(R2);
1C A2 80 7D 002D 278 MOVQ (R0)+,SCS$T-DST-PROC+8(R2);
0031 279 ;
0031 280 ; CLUSTER to report process names
50 1C A3 D0 0031 281 MOVL CDT$L_PB(R3),R0 ; for incomplete connect calls
34 A0 6C A3 D0 0035 282 MOVL PB$L_CDTLST(R0),- ; Get path block addr for CDT
34 A0 53 D0 003A 283 CDT$L_CDTLST(R3) ; Link this new CDT onto
07 B0 003E 284 R3,PB$L_CDTLST(R0) ; the head of the CDT list
28 A3 0040 285 MOVW #CDT$C_CON_SENT,- ; for this path
50 01 3C 0042 286 CDT$W_STATE(R3) ; Move CDT state to
69 11 0045 287 MOVZWL #CDT$C_CON_PEND,R0 ; connect sent
0047 288 BRB SCSEND ; Get block state
0047 289 ; Ask to send CONNECT_REQ & suspend
0047 290 CON_MEM_FAIL:
0047 291
50 DD 0047 292 PUSHL R0 ; Save error status
0049 293
0049 294 CON_MEM_FAIL1:
0049 295
00000000'GF 16 0049 296 JSB G^SCS$DEALL_CDT ; Deallocate CDT
50 8ED0 004F 297 POPL R0 ; Retrieve status
05 0052 298 RSB ; Return error to SYSAP
0053 299
0053 300 .DSABL LSB
```



```
0053 302 .SBTTL - FPC$ACCEPT, COMPLETE PROCESSING AN ACCEPT
0053 303
0053 304 :+
0053 305 : This routine is JMP'ed to by SCSS$ACCEPT which allocates and inits
0053 306 : a CDT on which the connection is to be completed. FPC$ACCEPT
0053 307 : allocates the SCS receive buffer, message buffers, and datagram
0053 308 : buffers the new connection will need and requests the SCS send
0053 309 : process to send an ACCEPT_REQ message to the remote system.
0053 310 : Finally, the SYSAP is suspended until the ACCEPT_RSP is received.
0053 311 :
0053 312 : Inputs:
0053 313 :
0053 314 : R2 -Addr of listening CDT
0053 315 : R3 -Addr of accepting CDT
0053 316 : R4 -Addr of PDT
0053 317 :
0053 318 : Listening CDT:
0053 319 :
0053 320 : CDT$W_STATE -Connect received state
0053 321 : CDT$L_SCSMSG -Addr of message buffer containing CONNECT_REQ
0053 322 : CDT$L_PB -Path Blk of connect request
0053 323 : CDT$B_RSTATION -Remote station addr of connect req
0053 324 : CDT$L_PDT -PDT of connect request
0053 325 :
0053 326 : Accepting CDT:
0053 327 : -All fields zeroed except:
0053 328 : MSGINPUT,DGINPUT,ERRADDR,MINSENT,
0053 329 : INITLREC, and DGREC as specified
0053 330 : by SYSAP;
0053 331 : LCONID, SIZE, TYPE, SUBTYP, WAITQFL
0053 332 : and WAITQBL
0053 333 : Outputs (upon resumption of SYSAP):
0053 334 :
0053 335 : R0 -Status: SS$_NORMAL, SS$_INSFMEM
0053 336 : R1 -Destroyed
0053 337 : R2 -Preserved if R0/SS$_INSFMEM; Else destroyed
0053 338 : Other registers -Preserved
0053 339 :
0053 340 : Listening CDT:
0053 341 :
0053 342 : CDT$W_STATE -LISTEN state
0053 343 :
0053 344 : Accepting CDT:
0053 345 : -All fields initialized
0053 346 : -
0053 347 :
0053 348 :
0053 349 : CDT adjacency assumptions:
0053 350 :
0053 351 :
0053 352 ASSUME CDT$L_PB+4 EQ CDT$B_RSTATION
0053 353
0053 354 .ENABL LSB
0053 355
0053 356 FPC$ACCEPT::
0053 357
0053 358 $CHK_CDTSTATE - ; Verify that accepting CDT
```



```
0053 359
0053 360
10 A3 54 DO 005C 361      MOVL      CLOSED,-
      1C A2 7D 0060 362      ERROR=$STATE_ERR
      1C A3      0063 363      R4,CDT$L_PDT(R3)
      24 A2 BO 0065 364      MOVQ      CDT$L_PB(R2),-
      24 A3      0068 365      CDT$L_PB(R3)
50 1C A2 DO 006A 366      MOVW      CDT$B_RSTATION+4(R2),-
      1C A2      006E 367      CDT$B_RSTATION+4(R3)
      34 A0 DO 006E 368      MOVL      CDT$L_PB(R2),R0
      6C A3      0071 369      MOVL      PBS$L_CDTLST(R0),-
34 A0 53 DO 0073 370      CDT$L_CDTLST(R3)
50 52 DO 0077 371      R3,PBS$L_CDTLST(R0)
      2C A2 DO 007A 372      MOVL      R2,R0
      2C A0 D4 007E 373      MOVL      CDT$L_SCSMSG(R2),R2
F8 A2 50 DO 0081 374      CLRL      CDT$L_SCSMSG(R0)
      50 DD 0085 375      MOVL      R0,SCS$L_DST_CONID(R2)
2C A3 52 DO 0087 376      PUSHL     R0
      FF72' 30 008B 377      MOVL      R2,CDT$L_SCSMSG(R3)
      008E 378      BSBW      SCS$COPY_ACCP
      14 A2 DE 008E 379      MOVAL     SCS$T_SRC_PROC(R2),-
      50 A3      0091 380      CDT$L_RPROCNAM(R3)
      04 A2 DE 0093 381      MOVAL     SCS$T_DST_PROC(R2),-
      54 A3      0096 382      CDT$L_LPROCNAM(R3)
      FF65' 30 0098 383      BSBW      SCS$ACL_ALLBUF2
      52 8ED0 009B 384      POPL      R2
      08 50 E8 009E 385      BLBS      R0,10$
      50 DD 00A1 386      PUSHL     R0
      FF5A' 30 00A3 387      BSBW      SCS$DEAL_SCSREC
      FFA0 31 00A6 388      BRW       CON_MEM_FAIL1
      0A BO 00A9 389      BRW       CON_MEM_FAIL1
      28 A3 00AB 390 10$:  MOVW      #CDT$C_ACCP_SENT,-
50 02 3C 00AD 391      CDT$W_STATE(R3)
      00B0 392      MOVZWL     #CDT$C_ACCP_PEND,R0
      00B0 393
      00B0 394 SCSSEND:
      FF4D' 30 00B0 395      BSBW      SCS$REQ_SCSSEND
      04B2 31 00B3 396      BRW       SUSP_CONCALL
      00B6 397
      00B6 398      .DSABL     LSB
```

state is closed; if not,
caller made error
Set PDT addr in accepting CDT
Copy from listener CDT to accepting:
PB addr, remote station, l.o.,
remote station, h.o. 2 bytes
Get path blk addr of connect
request that was saved in listener
Link the new CDT to the
head of the CDT list
for this path
Save listening CDT addr temporarily
Get addr of CONNECT_REQ msg
Zero listener scs rcv buffer addr
Save listening CDT addr in msg
and save on stack also
Put msg addr in accepting CDT
Copy credit, RCONID info from
CONNECT_REQ to accepting CDT
Set addr of remote proc name
and local proc name in CDT
for later xmit of ACCEPT_REQ
Allocate all msg and dg buffers
Retrieve listener CDT address
Branch if got them
Else save error status
Deallocate extra SCS rcv buffer
Clean up accepting CDT (status on stack)
Move state to accept sent
Set block state to accept pending
Ask to send ACCEPT_REQ
Suspend SYSAP connection call

- FPC\$REJECT, PROCESS A REJECT CALL

```
00B6 400      .SBTTL -      FPC$REJECT,      PROCESS A REJECT CALL
00B6 401
00B6 402      ;+
00B6 403      ; FPC$REJECT is called directly from the SYSAP. It requests
00B6 404      ; the SCS send process to send a REJECT_REQ message with SYSAP-
00B6 405      ; specified reject reason. FPC$REJECT then suspends the SYSAP
00B6 406      ; until the reject response arrives.
00B6 407
00B6 408      Inputs:
00B6 409
00B6 410      R0              -Reject reason (l.o. 16 bits)
00B6 411      R3              -Addr of CDT (listening CDT)
00B6 412      R4              -Addr of PDT
00B6 413
00B6 414      CDT$L_SCSMSG      -Addr of msg buffer containing CONNECT_REQ
00B6 415
00B6 416      Outputs (upon resumption of SYSAP):
00B6 417
00B6 418      R0              -SS$_NORMAL, SS$_ILLCDTST
00B6 419      R1,R2          -Destroyed
00B6 420      other registers -Preserved
00B6 421
00B6 422      CDT$W_STATE(R3)    -Connect rec'd --> listen
00B6 423      :-
00B6 424
00B6 425      .ENABL  LSB
00B6 426
00B6 427 FPC$REJECT::
00B6 428
00B6 429      $CHK_CDTSTATE -      ; Verify CDT state is
00B6 430      CON REC,-          ; connect received; if not,
00B6 431      ERROR=STATE ERR    ; caller made error
00B6 432      BSBW  SCSSMAP VMSSTS ; Map VMS status to SCS
00B6 433      MOVW  R0,CDT$W_REASON(R3) ; Save reject reason
00B6 434      MOVW  #CDT$C_REJ_SENT,- ; Move CDT state to reject sent
00B6 435      CDT$W_STATE(R3)
00B6 436      MOVZWL #CDT$C_REJ_PEND,R0 ; Set block state = reject pending
00B6 437      BRB   SCSSEND      ; Ask to send REJECT_REQ & suspend
00B6 438
00B6 439      .DSABL  LSB
```

26 A3 FF3E' 30 00BF 432
50 0B B0 00C2 433
28 A3 B0 00C6 434
50 03 3C 00C8 435
E1 11 00CA 436
00CD 437
00CF 438
00CF 439

- FPC\$DCONNECT, PROCESS A DISCONNECT CAL

```
.SBTTL - FPC$DCONNECT, PROCESS A DISCONNECT CALL

OOCF 441 :+ FPC$DCONNECT is called by the SYSAP. It may be called from
OOCF 442 : three states. Depending upon the state, the following actions
OOCF 443 : are taken:
OOCF 444 :
OOCF 445 :
OOCF 446 :
OOCF 447 :
OOCF 448 : STATE ACTIONS NEW STATE
OOCF 449 :
OOCF 450 : CLOSED No action; return success to the SYSAP,
OOCF 451 : $$$ALRDYCLOSED.
OOCF 452 :
OOCF 453 : OPEN Trade DISCONNECT's with the remote SYSAP.
OOCF 454 : When the trade is done, return success to
OOCF 455 : the SYSAP. The state changes seen by the
OOCF 456 : side initiating the DISCONNECT are:
OOCF 457 : OPEN-->DISC_SENT-->DISC_ACK-->CLOSED.
OOCF 458 : The state changes seen by the passive side are:
OOCF 459 : OPEN-->DISC_REC-->DISC_MTCH-->CLOSED.
OOCF 460 : If both sides initiate a DISCONNECT
OOCF 461 : simultaneously, so that the requests cross
OOCF 462 : in the mail, then each side sees the
OOCF 463 : following state transitions:
OOCF 464 : OPEN-->DISC_SENT-->DISC_MTCH-->CLOSED.
OOCF 465 :
OOCF 466 : CON_ACK, Unilaterally deallocate CDT and associated
OOCF 467 : DISC_ACK receive buffers. Complete original
OOCF 468 : outstanding CONNECT/DISCONNECT with abort
OOCF 469 : status, $$$ABORT. Return success on the
OOCF 470 : DISCONNECT call.
OOCF 471 :
OOCF 472 : CON_REC Do a REJECT.
OOCF 473 :
OOCF 474 : DISC_REC Send out a DISCONNECT (part of the normal
OOCF 475 : handshake discussed for OPEN.) The
OOCF 476 : DISCONNECT request is sent on the lowest
OOCF 477 : priority queue to delay it till all other
OOCF 478 : pending traffic, including block transfers,
OOCF 479 : is done. A credit message is forced out
OOCF 480 : first in order to make sure the remote
OOCF 481 : knows about all the credits we have to extend.
OOCF 482 :
OOCF 483 : Other states All other states represent the window
OOCF 484 : between sending an SCS request and getting
OOCF 485 : the response. During this window the CDT
OOCF 486 : cannot be unilaterally destroyed and so
OOCF 487 : error status $$$ILLCDTST is returned to
OOCF 488 : the SYSAP.
OOCF 489 :
OOCF 490 : Inputs:
OOCF 491 :
OOCF 492 : R0 -Disconnect reason
OOCF 493 : R3 -Addr of CDT being disconnected
OOCF 494 : R4 -Addr of PDT
OOCF 495 :
OOCF 496 : Outputs:
OOCF 497 :
```



```
- FPC$DCONNECT, PROCESS A DISCONNECT CAL

00CF 498 : R0 -Status: SSS_NORMAL, SSS_ILLCDTST
00CF 499 : R1,R2,R3 -Destroyed
00CF 500 : Other registers -Preserved
00CF 501 :-
00CF 502
00CF 503 .ENABL LSB
00CF 504
00CF 505 FPC$DCONNECT::
00CF 506
51 1C A3 D0 00CF 507 MOVL CDT$L_PB(R3),R1 ; Get PB addr
12 A1 B1 00D3 508 CMPW PBSW_STATE(R1),- ; Is path in either
8000 8F 03 12 00D6 509 #PBSVC_VC_FAIL ; virtual circuit fail or
FF22' 31 00D9 510 BNEQ 2$ ;
00DB 511 BRW SCSS$DISC_VCFAIL ;
00DE 512
12 A1 B1 00DE 513 2$: CMPW PBSW_STATE(R1),- ; power fail state?
4000 8F 03 12 00E1 514 #PBSVC_PWR_FAIL ;
00E4 515 BNEQ 3$ ;
FF17' 31 00E6 516 BRW ERR$DISC_PWFAIL ; If so, call different DISCONNECT
00E9 517
00E9 518 3$: $DISPATCH -
00E9 519 CDT$W_STATE(R3),- ; Dispatch on CDT state:
00E9 520 <- (CLOSED/LISTEN handled by SCSLOA)
00E9 521 <CDT$C_OPEN, DISC_OPEN>,- ; OPEN,
00E9 522 <CDT$C_CON_ACK, DISC_CON_ACK>,- ; CON_ACK,
00E9 523 <CDT$C_DISC_ACK,DISC_ILLSTATE>,- ; DISC_ACK,
00E9 524 <CDT$C_CON_REC, FPC$REJECT>,- ; CON_REC,
00E9 525 <CDT$C_DISC_REC,DISC_DISC_REC>,- ; DISC_REC,
00E9 526 <CDT$C_CON_SENT,DISC_ILLSTATE>,- ; CON_SENT,
00E9 527 <CDT$C_DISC_SENT,DISC_ILLSTATE>,- ; DISC_SENT,
00E9 528 <CDT$C_REJ_SENT,DISC_ILLSTATE>,- ; REJ_SENT,
00E9 529 <CDT$C_ACCP_SENT,DISC_ILLSTATE>,- ; ACCP_SENT
00E9 530 <CDT$C_DISC_MTCH,DISC_ILLSTATE>,- ; Matching DISC sent
00E9 531 > ; (CDT$C_VC_FAIL went to SCSS$DISC
0102 532
0102 533 BUGCHECK CIPT, NONFATAL ; If none of the above
0109 534 ; states, system error,
0109 535 ; possibly recoverable
50 01 3C 0109 536 MOVZWL #SS$_NORMAL,R0 ; If bugcheck nonfatal, return
05 010C 537 RSB ; success to SYSAP
010D 538
010D 539 ;
010D 540 ; Connection can't be closed right now without violating SCS protocol.
010D 541 ; Therefore close unilaterally and crash VC.
010D 542 ;
010D 543
010D 544 DISC_ILLSTATE:
010D 545
1C A3 DD 010D 546 PUSHL CDT$L_PB(R3) ; Save PB addr
0A 10 0110 547 BSBB DISC_CON_ACK ; Cleanup CDT and pending
51 8ED0 0112 548 ; CONNECT/DISCONNECT
FEE8' 30 0112 549 POPL R1 ; Retrieve PB address
50 01 3C 0115 550 BSBW ERR$CRASHVC ; Initiate VC crash
0118 551 MOVZWL #SS$_NORMAL,R0 ; Set status to return to caller
0118 552 ; on latest DISCONNECT call
05 0118 553 RSB ; Return error to SYSAP
011C 554
```



```
011C 555 :  
011C 556 : CDT has a CONNECT or DISCONNECT request already pending. Unilaterally  
011C 557 : clean up the CDT. Complete pending request with abort status.  
011C 558 : Complete this DISCONNECT with success.  
011C 559 :  
011C 560  
011C 561 DISC_CON_ACK:  
011C 562  
011C 563      MOVQ      R4,-(SP)                ; Save R4, R5  
55 7E 54 7D 011C 564      MOVL      CDT$R5(R3),R5          ; Restore context from pending  
68 A3 DO 011F 565      PUSHL     CDT$R5(R3)              ; connx mgmt call  
64 A3 DD 0123 566      BSBW      SCSS$DEAL_ALLBUF        ; Clean up all receive buffers  
FED7' 30 0126 567      JSB       G*SCSS$DEALL_CDT        ; Deallocate CDT (close status)  
00000000'GF 16 0129 568      MOVZWL   #SS$ABORT,R0        ; Set status to abort  
50 2C 3C 012F 569      JSB       @($P)+                ; Restore pending call thread  
9E 16 0132 570      MOVQ      (SP)+,R4                ; Restore R4, R5  
54 8E 7D 0134 571      MOVZWL   #SS$_NORMAL,R0          ; Set DISCONNECT status to ok  
50 01 3C 0137 572      RSB                    ; Return from DISCONNECT  
05 013A 573  
013B 574 :  
013B 575 : SYSAP has received an unsolicited DISCONNECT request from the  
013B 576 : remote SYSAP and now wishes to issue the matching DISCONNECT.  
013B 577 :  
013B 578  
013B 579 DISC_DISC_REC:  
013B 580  
26 A3 FEC2' 30 013B 581      BSBW      SCSS$MAP_VMSSTS        ; Convert reason from VMS to SCS  
50 50 80 013E 582      MOVW      R0,CDT$R5_REASON(R3)      ; Save DISCONNECT reason  
06 80 0142 583      MOVW      #CDT$C_DISC_MTCH,-          ; Move CDT state to  
28 A3 0144 584      CDT$R5_STATE(R3)                    ; matching DISCONNECT sent  
50 04 3C 0146 585      MOVZWL   #CDT$C_DISC_PEND,R0        ; Set block state to send DISCONNECT  
FEB4' 30 0149 586 10$: BSBW      SCSS$REQ_SCSSSEND        ; Send out the DISCONNECT  
0419 31 014C 587 20$: BRW      SUSP_CONCALL              ; Suspend SYSAP till done  
014F 588  
014F 589 :  
014F 590 : Connection is OPEN. Force sending of any unextended credits (may  
014F 591 : send 0 credits). Send out a DISCONNECT on the lowest priority queue.  
014F 592 : Move CDT state from OPEN to DISC_SENT.  
014F 593 :  
014F 594  
014F 595 DISC_OPEN:  
014F 596  
26 A3 FEAE' 30 014F 597      BSBW      SCSS$MAP_VMSSTS        ; Convert status to SCS  
50 50 80 0152 598      MOVW      R0,CDT$R5_REASON(R3)      ; Save DISCONNECT reason  
05 80 0156 599      MOVW      #CDT$C_DISC_SENT,-          ; Set CDT state to  
28 A3 0158 600      CDT$R5_STATE(R3)                    ; show DISCONNECT sent  
50 06 80 015A 601      MOVW      #CDT$C_DCR_PEND,R0        ; Block state will be disconnect  
015D 602      + credit pending  
EA 11 015D 603      BRB       10$                          ; Request SCS send and suspend  
015F 604      SYSAP till DISCONNECT complete  
015F 605  
015F 606      .DSABL   LSB
```


SEQUENCED MESSAGE CALLS

```
015F 608 .SBTTL SEQUENCED MESSAGE CALLS
015F 609 .SBTTL - FPC$ALLOCMSG, ALLOCATE A MESSAGE BUFFER
015F 610
015F 611 :+
015F 612 : FPC$ALLOCMSG checks if there is at least one send credit. If not, the
015F 613 : SYSAP is suspended behind other waiting SYSAP's until there is. The
015F 614 : message buffer is allocated from nonpaged pool. If insufficient pool
015F 615 : is available, then the SYSAP is suspended until pool is available.
015F 616 : The destination connection ID is then copied to the SCS header
015F 617 : at this time so that the message can be sent harmlessly even if
015F 618 : a power failure should occur. (It will be discarded at the receiving
015F 619 : end upon detection of connect ID sequence number mismatch.) Finally,
015F 620 : the address of the start of the application data within the buffer is
015F 621 : computed and returned to the SYSAP.
015F 622
015F 623 : Inputs:
015F 624
015F 625 : R4 -Addr of PDT
015F 626 : R5 -Addr of CDRP
015F 627 : CDRP$L_CDT -Addr of CDT
015F 628
015F 629 : Outputs:
015F 630
015F 631 : R0 -Status: SS$_NORMAL, SS$_ILLCDTST
015F 632 : R1 -Destroyed
015F 633 : R2 -Addr of message buffer, if status=success
015F 634 : Other registers -Preserved
015F 635
015F 636 : CDRP$L_MSG_BUF -Addr of message buffer, if status=success
015F 637 :-
015F 638
015F 639 .ENABL LSB
015F 640
015F 641 FPC$ALLOCMSG::
015F 642
51 24 A5 D0 015F 643 MOVL CDRP$L_CDT(R5),R1 : Get CDT addr
0163 644 $CHK_CDTSTATE - : Verify connection state
0163 645 OPEN,- : is open.
0163 646 ERROR=STATE_ERR,- : Else report error to SYSAP
0163 647 CDT=R1 :
18 A5 8ED0 016C 648 POPL CDRP$L_SAVD RTN(R5) : Save 1st level return
40 A1 B5 0170 649 TSTW CDT$W_SEND(R1) : Got any credit for send?
1D 1A 0173 650 BGTRU 10$ : Branch if so
0098 C1 B6 0175 651 INCW CDT$W_QCR_CNT(R1) : Step count of # credit waits
0179 652 $$SUSP_SCS - : Else suspend SCS routine
0179 653 @CDT$L_CRWAITQBL(R1) : on credit wait queue
0192 654
FE6B' 30 0192 655 10$: BSBW INT$ALLOC_MSG : Allocate a message buffer
1C 50 E8 0195 656 BLBS R0,20$ : Branch if got it
0198 657 $$SUSP_SCS - : Else suspend this routine
0198 658 @PDT$L_WAITQBL(R4) : on pool wait queue
DE 11 01B2 659 BRB 10$ : Try to allocate now
01B4 660
51 24 A5 D0 01B4 661 20$: MOVL CDRP$L_CDT(R5),R1 : Get CDT addr again
14 A1 D0 01B8 662 MOVL CDT$L_RCONID(R1),- : Set destination connect
F8 A2 01BB 663 SCSSL_DST_CONID(R2) : ID in SCS header
1C A5 52 D0 01BD 664 MOVL R2,CDRP$L_MSG_BUF(R5) : data and save in CDRP
```


PAFPCALL
V04-001

J 2

- FPC\$ALLOCMSG, ALLOCATE A MESSAGE BUFFE 16-SEP-1984 01:10:45 VAX/VMS Macro V04-00
10-SEP-1984 01:15:44 [DRIVER.SRC]PAFPCALL.MAR;2

Page 15
(8)

4C 11 01C1 665

BRB 50\$

; Join common exit code

- FPC\$RCHMSGBUF, RECYCLE MESSAGE BUFFER

```
01C3 667 .SBTTL - FPC$RCHMSGBUF, RECYCLE MESSAGE BUFFER
01C3 668 .SBTTL - AT HIGH PRIORITY
01C3 669 .SBTTL - FPC$RCLMSGBUF, RECYCLE MESSAGE BUFFER
01C3 670 .SBTTL - AT LOW PRIORITY
01C3 671
01C3 672
01C3 673 :+ FPC$SRCxMSGBUF checks if there is at least one send credit. If
01C3 674 : not, the SYSAP is suspended until there is. FPC$SRCxMSGBUF then
01C3 675 : decrements the send credit. The wait, if required, places the
01C3 676 : SYSAP CDRP at the end of the wait queue for low priority and at
01C3 677 : the head of the queue for high priority. The address of the
01C3 678 : buffer being recycled is returned in both R2 and CDRP$MSG_BUF.
01C3 679 : The remote connection ID is set in the SCS header so that the
01C3 680 : message can be sent harmlessly even if a power failure should occur.
01C3 681 : (It will be discarded by the receiving SCS.)
01C3 682
01C3 683 : Inputs:
01C3 684
01C3 685 : R4 -Addr of PDT
01C3 686 : R5 -Addr of CDRP
01C3 687 : CDRP$CDT -Addr of CDT
01C3 688 : CDRP$MSG_BUF -Addr of msg buffer
01C3 689
01C3 690 : Outputs:
01C3 691
01C3 692 : R0 -Status: SSS_NORMAL, SSS_ILLCDTST
01C3 693 : R2 -Addr of message buffer
01C3 694 : R1 -Destroyed
01C3 695 : Other registers -Preserved
01C3 696 : CDRP$MSG_BUF(R5) -Addr of message buffer
01C3 697 :-
01C3 698
01C3 699
01C3 700 FPC$RCHMSGBUF::
01C3 701
51 24 A5 D0 01C3 702 MOVL CDRP$CDT(R5),R1 ; Get CDT addr
50 38 A1 DE 01C7 703 MOVAL CDT$RWAITQFL(R1),R0 ; Get addr of head of wait queue
08 11 01CB 704 BRB 30$ ; Join common processing
01CD 705
01CD 706 FPC$RCLMSGBUF::
01CD 707
51 24 A5 D0 01CD 708 MOVL CDRP$CDT(R5),R1 ; Get CDT addr
50 3C A1 D0 01D1 709 MOVL CDT$RWAITQBL(R1),R0 ; Get addr of end of wait queue
01D5 710
01D5 711 30$: $CHK_CDTSTATE - ; Verify connection state
01D5 712 OPEN,- ; is open
01D5 713 ERROR=STATE_ERR,- ; Else report error to SYSAP
01D5 714 CDT=R1 ;
01DE 715 POPL CDRP$SAVD_RTN(R5) ; Copy return to SYSAP from stack
01E2 716 ; to CDRP
01E2 717 TSTW CDT$W_SEND(R1) ; Got a send credit?
01E5 718 BGTRU 40$ ; Branch if so
51 50 D0 01E7 719 MOVL R0,R1 ; Get queue hdr in less volatile
01EA 720 ; register
01EA 721 $$SUSP_SCS (R1) ; Else suspend this routine
51 24 A5 D0 0202 722 MOVL CDRP$CDT(R5),R1 ; Retrieve CDT addr
0206 723
```


- AT LOW PRIORITY

52	1C	A5	D0	0206	724	40\$:	MOVL	CDRPSL MSG BUF(R5),R2	; Get msg addr in register
	14	A1	D0	020A	725		MOVL	CDTSL_RCONID(R1),-	; Set remote CONID in SCS header
	F8	A2		020D	726			SCSSL-DST CONID(R2)	
	40	A1	B7	020F	727	50\$:	DECW	CDTSL_SEND(R1)	; Mark one credit used
50		01	3C	0212	728		MOVZWL	#SS\$ NORMAL,R0	; Set status to success
	18	B5	17	0215	729		JMP	@CDRPSL_SAVD_RTN(R5)	; Return to SYSAP
				0218	730				
				0218	731		.DSABL	LSB	

- FPC\$DEALLOCMSG, DEALLOCATE A MESSAGE BU

```
0218 733 .SBTTL - FPC$DEALLOCMSG, DEALLOCATE A MESSAGE BUFFER
0218 734 .SBTTL - FPC$DEALRGMSG, DEALLOCATE A MESSAGE BUFFER,
0218 735 .SBTTL - ARGUMENTS PASSED IN REGISTERS
0218 736
0218 737 :+
0218 738 : FPC$DEALLOCMSG resets the message address specified by the caller to
0218 739 : the top of the message buffer and compares the current number of
0218 740 : receive message buffers with the initial count specified at the
0218 741 : time of the connect. If the current receive count is not less than
0218 742 : the initial, then the message buffer is deallocated to nonpaged
0218 743 : pool. If the current receive count is less than the initial,
0218 744 : then the buffer is added to the free message queue and the pending
0218 745 : receive count is incremented. If the receive count was also less
0218 746 : than the minimum required by the remote SYSAP plus the flow control
0218 747 : cushion (SCS$GW_FLOWCUSH), then the CDT is queued for sending a
0218 748 : CREDIT message to the remote.
0218 749
0218 750 : Entry FPC$DEALRGMSG is called with the same arguments as DEALLOMSG,
0218 751 : but in registers instead of the CDRP.
0218 752
0218 753 : Inputs:
0218 754
0218 755 : R2 -Addr of message buffer (FPC$DEALRGMSG)
0218 756 : R3 -Addr of CDT (FPC$DEALRGMSG)
0218 757 : R4 -Addr of PDT
0218 758 : R5 -Addr of CDRP
0218 759 : CDRP$L_CDT -Addr of CDT (FPC$DEALLOCMSG)
0218 760 : CDRP$L_MSG_BUF -Addr of msg buffer (FPC$DEALLOCMSG)
0218 761
0218 762 : Outputs:
0218 763
0218 764 : R0-R2 -Destroyed
0218 765 : Other registers -Preserved
0218 766 : CDRP$L_MSG_BUF(R5) -0 (FPC$DEALLOCMSG only)
0218 767 :-
0218 768
0218 769 .ENABL LSB
0218 770
0218 771 FPC$DEALLOCMSG::
0218 772
0218 773 : PUSH R3 : Save caller's R3
0218 774 : MOV CDRP$L_MSG_BUF(R5),R2 : Get addr of message buffer
0218 775 : MOV CDRP$L_CDT(R5),R3 : Get addr of CDT
0218 776 : BSBB FPC$DEALRGMSG : Call routine to deallocate
0218 777 : POPL R3 : Restore caller's R3
0218 778 : CLRL CDRP$L_MSG_BUF(R5) : Zero msg addr in CDRP
0218 779 : RSB : Return
0218 780
0218 781 FPC$DEALRGMSG::
0218 782 : Entry for appl data pointer in R2
0218 783 : and CDT addr in R3
0218 784 : ADDW3 CDT$W_REC(R3),- : Compute total receive credits now
0218 785 : CDT$W_PENDREC(R3),R0 : = extended + not yet extended
0218 786 : CMPW R0,CDT$W_INITLREC(R3) : Total receive less than initial?
0218 787 : BLSSU 10$ : Branch if so
0218 788 : BSBW INT$DEAL_MSG : Deallocate message buffer-
0218 789 : CLXL R2 : to nonpaged pool
0218 : :***Debug code*
```

52 1C A5 DD 0218 773
53 24 A5 D0 021A 774
07 10 0222 775
53 8ED0 0224 776
1C A5 D4 0227 777
05 022A 778
022B 779
022B 780
022B 781
50 42 A3 A1 022B 783
46 A3 022E 784
48 A3 50 B1 0231 785
06 1F 0235 786
FDC6' 30 0237 787
52 D4 023A 788
023A 789

- ARGUMENTS PASSED IN REGISTERS

	05	023C	790		RSB		; Return to SYSAP
		023D	791				
FDC0'	30	023D	792	10\$:	BSBW	INT\$INS MFREQ	; Insert buffer on free queue
46 A3	B6	0240	793		INCW	CDT\$W_PENDREC(R3)	; Reflect insert in credit
00000000'GF	A1	0243	794		ADDW3	G^SCS\$GW_FLOWCUSH,-	; Compute cushion + minimum #
50 44 A3		0249	795			CDT\$W_MINREC(R3),R0	; send credits req'd by remote
50 42 A3	B1	024C	796		CMPW	CDT\$W_REC(R3),R0	; Is current # rcv buffers less
		0250	797				; cushion + minimum?
13	1A	0250	798		BGTRU	30\$; Branch if not
50 05	3C	0252	799		MOVZWL	#CDT\$C_CR_PEND,R0	; Get credit block state code
		0255	800		\$DISPATCH -		
		0255	801			CDT\$W_STATE(R3),-	; If connection is in any of the
		0255	802			<-	; states that indicate
		0255	803			<CDT\$C_DISC_ACK,30\$>,-	; that the local SYSAP has
		0255	804			<CDT\$C_DISC_SENT,30\$>,-	; issued a DISCONNECT request,
		0255	805			<CDT\$C_DISC_MTCH,30\$>,-	; then don't ask to send a credit
		0255	806			>	; request -- if we are already
		0262	807				; queued to send the final credit
		0262	808				; that preceeds the DISCONNECT, then
		0262	809				; the latest credit will be included,
		0262	810				; otherwise it won't
		0262	811				
FD9B'	30	0262	812	20\$:	BSBW	SCS\$REQ_SCSEND	; Request xmit of credit message
		0265	813				
	05	0265	814	30\$:	RSB		; Return to SYSAP
		0266	815				
		0266	816		.DSABL	LSB	


```
0266 818 .SBTTL - FPC$SENDMSG, SEND A SEQUENCED MESSAGE
0266 819
0266 820 ;+
0266 821 ; The SCS header of the specified message buffer is filled in.
0266 822 ; If the response ID is 0, then the message is queued for transmission
0266 823 ; with RETFLAG = 1(TRUE) thus channeling the sent buffer to the response
0266 824 ; queue for reclaim. If the response ID is non-zero, then a
0266 825 ; response is expected from the remote SYSAP and the message is sent
0266 826 ; with RETFLAG = 0(FALSE). RETFLAG = FALSE channels the sent buffer to
0266 827 ; the message free queue in anticipation of the response. In this
0266 828 ; case the receive credit is also incremented to account for the
0266 829 ; buffer being added to the free queue. All messages are sent
0266 830 ; on the high priority queue.
0266 831
0266 832 Inputs:
0266 833
0266 834 R1 -# bytes application data (FPC$SENDMSG)
0266 835 R4 -Addr of PDT
0266 836 R5 -Addr of CDRP
0266 837 CDRP$L_CDT(R5) -Addr of CDT
0266 838 CDRP$L_MSG_BUF(R5) -Addr of message
0266 839 CDRP$L_RSPID(R5) -RSPID (to set RETFLG)
0266 840
0266 841 Outputs:
0266 842
0266 843 R0 -Status: SSS_NORMAL, SSS_ILLCDTST
0266 844 R1,R2 -Destroyed
0266 845 Other registers -Preserved
0266 846
0266 847 CDRP$L_MSG_BUF(R5) -Zeroed to show msg buffer gone
0266 848 :-
0266 849
0266 850 .ENABL LSB
0266 851
0266 852 FPC$SENDMSG::
0266 853
51 00000000'GF 3C 0266 854 MOVZWL G^SCS$GW_MAXMSG,R1 ; Set for default
0266 855
0266 856 FPC$SENDMSG::
0266 857
0266 858 PUSHL R3 ; Save caller's R3
53 24 A5 DD 0266 859 MOVL CDRP$L_CDT(R5),R3 ; Get CDT addr in R3
0266 860 $CHK_CDTSTATE - ; Verify connection is
0266 861 OPEN,- ; open
0266 862 ERROR=STATE_ERR_R3,- ; Else report error to SYSAP
0266 863 CDT=R3 ;
0266 864 CLRL R0 ; Assume RETFLAG will be false
0266 865 ; and we will put msg on free queue
0266 866 TSTL CDRP$L_RSPID(R5) ; Is there a rspid?
0266 867 BNEQ 10$ ; Branch if there is
0266 868 ADDW3 CDT$W_REC(R3),- ; Else compute total receive credits
0266 869 CDT$W_REC(R3),R2 ; queued now
0266 870 CMPW R2,CDT$W_INITLREC(R3) ; Current rcv less than initial?
0266 871 BLSSU 10$ ; Branch if so
0266 872 MOVL #SYSAP$C_DISPRET,R0 ; Else set RETFLAG true
0266 873 BRB 20$ ; Join common processing
0266 874
```



```
- FPC$SENDMSG, SEND A SEQUENCED MESSAGE

      46 A3 B6 0294 875 10$: INCW CDT$W_PENDREC(R3) ; Step pending receive to reflect
      0297 876 ; msg port will put on free queue
      0297 877
      0297 878
      52 1C A5 D0 0297 879 20$: MOVL CDRP$L_MSG_BUF(R5),R2 ; Get message buffer addr
FO A2 51 0E A1 0298 880 ADDW3 #SCS$C_CVHD,R1,- ; Set SCS length
      02A0 881 SCS$W_LENGTH(R2)
      0A B0 02A0 882 MOVW #SCS$C_APPL_MSG,- ; Set SCS type to application
      F4 A2 02A2 883 SCS$W_MTYPE(R2) message
      46 A3 B0 02A4 884 MOVW CDT$W_PENDREC(R3),- ; Extend any pending receive
      F6 A2 02A7 885 SCS$W_CREDIT(R2) credits to the remote
      46 A3 A0 02A9 886 ADDW CDT$W_PENDREC(R3),- ; Move pending receives to
      42 A3 02AC 887 CDT$W_REC(R3) actual receives (real send
      02AE 888 ; credits extended)
      46 A3 B4 02AE 889 CLRW CDT$W_PENDREC(R3) ; No more pending credit
      18 A3 D0 02B1 890 MOVL CDT$L_LCONID(R3),- ; Put local connection ID
      FC A2 02B4 891 SCS$L_SRC_CONID(R2) into header
      51 1C A3 D0 02B6 892 MOVL CDT$L_PB(R3),R1 ; Get address of PB in R1
      7C A3 D6 02BA 893 INCL CDT$L_MSGSENT(R3) ; Step count of msgs sent
      FD40' 30 02BD 894 BSBW INT$SENDMSG ; Send the message with RETFLAG in R0
      53 8ED0 02C0 895 POPL R3 ; Restore SYSAP's R3
      1C A5 D4 02C3 896 CLRL CDRP$L_MSG_BUF(R5) ; Mark msg as no longer held by CDRP
      20 A5 D5 02C6 897 TSTL CDRP$L_RSPID(R5) ; Was RETFLAG true?
      13 13 02C9 898 BEQL FPC_SUCCESS ; Branch if yes
      02CB 899 $SUSP_FP ; Save fork process' context
      02D4 900
      02D4 901 .DSABL LSB
```


DATAGRAM SERVICE CALLS

```
02D4 903      .SBTTL  DATAGRAM SERVICE CALLS
02D4 904      .SBTTL  -      FPC$ALLOCDG,    ALLOCATE A DATAGRAM BUFFER
02D4 905
02D4 906      :+
02D4 907      : FPC$ALLOCDG allocates one datagram buffer from nonpaged pool.  If
02D4 908      : none is available, error status is returned to the caller.  Otherwise,
02D4 909      : the address of space for application data within the buffer
02D4 910      : is computed and returned to the caller.
02D4 911      :
02D4 912      : Inputs:
02D4 913      :
02D4 914      :      R4      -Addr of PDT
02D4 915      :      R5      -Addr of CDRP
02D4 916      :
02D4 917      : Outputs:
02D4 918      :
02D4 919      :      R0      -Status:  SS$ NORMAL, SS$ INSFMEM
02D4 920      :      R2      -Addr of dg, start of application data
02D4 921      :      CDRP$L_MSG_BUF  -Copy of R2
02D4 922      :      Other registers  -Preserved
02D4 923      :-
02D4 924
02D4 925      .ENABL  LSB
02D4 926
02D4 927  FPC$ALLOCDG::
02D4 928
02D4 929      BSBW  INT$ALLOC_DG      ; Allocate 1 dg buffer from pool
02D7 930      BLBC  R0,DG_ALC_FAIL  ; Branch if failed
02DA 931      MOVL  R2,CDRP$L_MSG_BUF(R5) ; Save addr in CDRP
02DE 932
02DE 933  FPC_SUCCESS:
02DE 934
02DE 935      MOVZWL  #SS$_NORMAL,R0      ; Set status to success
02E1 936      RSB                                ; Return
02E2 937
02E2 938  DG_ALC_FAIL:
02E2 939
02E2 940      MOVZWL  #SS$_INSFMEM,R0      ; Set status to failure
02E7 941      RSB                                ; Return
02E8 942
02E8 943      .DSABL  LSB
```

FD29' 30 02D4 929 BSBW INT\$ALLOC_DG ; Allocate 1 dg buffer from pool
08 50 E9 02D7 930 BLBC R0,DG_ALC_FAIL ; Branch if failed
1C A5 52 D0 02DA 931 MOVL R2,CDRP\$L_MSG_BUF(R5) ; Save addr in CDRP
50 01 3C 02DE 933 FPC_SUCCESS:
05 02DE 934
02DE 935 MOVZWL #SS\$_NORMAL,R0 ; Set status to success
02E1 936 RSB ; Return
02E2 937
02E2 938 DG_ALC_FAIL:
50 0124 8F 3C 02E2 939
05 02E2 940 MOVZWL #SS\$_INSFMEM,R0 ; Set status to failure
02E7 941 RSB ; Return
02E8 942
02E8 943 .DSABL LSB


```
- FPC$DEALLOCDG, DEALLOCATE A DATAGRAM B
02E8 945 .SBTTL - FPC$DEALLOCDG, DEALLOCATE A DATAGRAM BUFFER
02E8 946 .SBTTL - TO NONPAGED POOL
02E8 947
02E8 948 :+
02E8 949 : FPC$DEALLOCDG simply converts the datagram address to the address
02E8 950 : of the start of the buffer containing the datagram and calls
02E8 951 : COM$DRVDEALMEM.
02E8 952 :
02E8 953 : Inputs:
02E8 954 :
02E8 955 : R2 -Addr of datagram
02E8 956 : R4 -Addr of PDT
02E8 957 :
02E8 958 : Outputs:
02E8 959 :
02E8 960 : R0 -Destroyed
02E8 961 : R2 -0
02E8 962 : Other registers -Preserved
02E8 963 :-
02E8 964
02E8 965 .ENABL LSB
02E8 966
02E8 967 FPC$DEALLOCDG::
02E8 968
FD15' 30 02E8 969 BSBW INT$DEAL_DG ; Deallocate buffer
52 D4 02E8 970 CLRL R2 ; Mark dg addr as gone
05 02ED 971 RSB ; Return to SYSAP
02EE 972
02EE 973 .DSABL LSB
```


- FPC\$QUEUEDG, QUEUE A SYSAP SUPPLIED BU

```
02EE 975      .SBTTL -      FPC$QUEUEDG,      QUEUE A SYSAP SUPPLIED BUFFER
02EE 976      .SBTTL -      TO THE DATAGRAM FREE QUEUE
02EE 977
02EE 978      :+
02EE 979      : FPC$QUEUE allows a SYSAP to supply the port with a buffer to insert
02EE 980      : on the datagram free queue. The SYSAP must correctly set the type
02EE 981      : and size field before calling this routine. The datagram receive
02EE 982      : count in the SYSAP's CDT is incremented.
02EE 983
02EE 984      : INPUTS:
02EE 985
02EE 986      :      R2      -Addr of start of buffer (NOT appl data)
02EE 987      :      R3      -Addr of CDT
02EE 988      :      R4      -Addr of PDT
02EE 989      :      CDT$W_DGREC(R3) -Current DG receive count
02EE 990
02EE 991      : OUTPUTS:
02EE 992
02EE 993      :      R0      -Status: SSS_NORMAL
02EE 994      :      R1      -Preserved
02EE 995      :      R2      -Zeroed
02EE 996      :      CDT$W_DGREC(R3) -Incremented
02EE 997      :-
02EE 998
02EE 999 FPC$QUEUEDG::
02EE 1000
FDOF' 30 02EE 1001      BSBW      INT$INS_DFREQX      ; Insert buffer on port queue
4C A3 B6 02F1 1002      INCW      CDT$W_DGREC(R3)      ; Step SYSAP's receive count
20 11 02F4 1003      BRB      Q_SUCCESS      ; Finish up
```


- FPC\$QUEUEMDGS, ALLOCATE DG'S AND QUEUE

```

02F6 1005 .SBTTL - FPC$QUEUEMDGS, ALLOCATE DG'S AND QUEUE FOR
02F6 1006 .SBTTL - RECEIVES OR
02F6 1007 .SBTTL - DEQUEUE DG'S AND RETURN TO
02F6 1008 .SBTTL - NONPAGED POOL
02F6 1009
02F6 1010 :+
02F6 1011 : FPC$QUEUEMDGS is used by SYSAP's to alter the number of datagram buffers
02F6 1012 : they have queued for receives. The datagram count is positive if
02F6 1013 : datagrams are to be allocated from pool and queued for receives. The
02F6 1014 : count argument is negative if datagrams are to be removed from the queue
02F6 1015 : and returned to nonpaged pool.
02F6 1016 :
02F6 1017 : If datagrams are being added, then for each one allocated and queued,
02F6 1018 : the datagram receive count in the SYSAP's CDT is incremented. If there
02F6 1019 : is insufficient pool for all to be allocated, then the number actually
02F6 1020 : queued is returned to the SYSAP with a warning status.
02F6 1021 :
02F6 1022 : If datagrams are being withdrawn from the queue, then for each
02F6 1023 : one dequeued and returned to pool, the datagram receive count in the
02F6 1024 : SYSAP's CDT is decremented. If the datagram receive count reaches
02F6 1025 : 0 before all that the SYSAP requested have been dequeued, then the
02F6 1026 : number actually dequeued is returned to the caller with warning
02F6 1027 : status.
02F6 1028 :
02F6 1029 : Inputs:
02F6 1030 :
02F6 1031 : R1 -# of dg's to add (+) or
02F6 1032 : to withdraw (-)
02F6 1033 : R3 -Addr of CDT
02F6 1034 : R4 -Addr of PDT
02F6 1035 : CDT$W_DGREC(R3) -Current dg receive count
02F6 1036 :
02F6 1037 : Outputs:
02F6 1038 :
02F6 1039 : R0 -Status: SSS_NORMAL, SSS_DGQINCOMP
02F6 1040 : (Datagram queuing incomplete)
02F6 1041 : R1 -# actually added (+) or withdrawn (-)
02F6 1042 : R2 -Destroyed
02F6 1043 : Other registers -Preserved
02F6 1044 : CDT$W_DGREC(R3) -Updated
02F6 1045 :-
02F6 1046 :
02F6 1047 : .ENABL LSB
02F6 1048 :
02F6 1049 FPC$QUEUEMDGS::
02F6 1050 :
02F6 1051 CLRL -(SP) ; Set running dg count = 0
02F6 1052 TSTL R1 ; Check dg count requested
02F6 1053 BEQL Q_SUCCESS ; Branch if nothing to do
02F6 1054 BLSS DQUEUE_DG ; Branch if withdrawing
02F6 1055 :
02F6 1056 QUEUE_DG:
02F6 1057 :
02F6 1058 PUSHL R1 ; Save count argument
02F6 1059 BSBW INT$ALLOC_DG ; Allocate a dg buffer
02F6 1060 POPL R1 ; Restore argument
02F6 1061 BLBC R0,Q_INCOMPLETE ; Branch if allocate failed

```

```

7E D4
51 D5
1A 13
1C 19

51 DD
FCFD' 30
51 BED0
2D 50 E9

```



```

- NONPAGED POOL
FCF4' 30 0309 1062 BSBW INT$INS DFREQ ; Else insert buffer on port queue
4C A3 B6 030C 1063 INCW CDT$W DGREC(R3) ; Step SYSAP's receive count
EB 6E 51 F2 030F 1064 AOBLS R1,(SP),QUEUE_DG ; Step running tally and branch
; if less than requested
51 8ED0 0313 1065 POPL R1 ; Retrieve total tally from stack
0316 1066
0316 1067 Q_SUCCESS:
0316 1068
50 01 3C 0316 1070 MOVZWL #SS$_NORMAL,R0 ; Set status to success
05 0319 1071 RSB ; Return to SYSAP
031A 1072
031A 1073 DQUEUE_DG:
031A 1074
51 51 CE 031A 1075 MNEGL R1,R1 ; Trun request count positive
031D 1076
4C A3 B5 031D 1077 20$: TSTW CDT$W DGREC(R3) ; SYSAP have more dg's queued?
11 15 0320 1078 BLEQ DQ_INCOMPLETE ; Branch if not
FCDB' 30 0322 1079 BSBW INT$DFQ2POOL ; Remove a dg from free queue
OF 1D 0325 1080 BVS Q_INCOMPLETE ; Branch if none
4C A3 B7 0327 1081 DECW CDT$W DGREC(R3) ; Decrement SYSAP's recv count
EF 6E 51 F2 032A 1082 AOBLS R1,(SP),20$ ; Step running tally, branch
; if more to do
51 8E CE 032E 1084 MNEGL (SP)+,R1 ; Retrieve total tally and negate
E3 11 0331 1085 BRB Q_SUCCESS ; Join common success exit
0333 1086
0333 1087 DQ_INCOMPLETE:
0333 1088
6E 6E CE 0333 1089 MNEGL (SP),(SP) ; Turn tally into negative #
0336 1090
0336 1091 Q_INCOMPLETE:
0336 1092
51 8ED0 0336 1093 POPL R1 ; Retrieve tally from stack
50 09C0 8F 3C 0339 1094 MOVZWL #SS$_DGQINCOMP,R0 ; Set status to error
05 033E 1095 RSB ; Return
033F 1096
033F 1097 .DSABL LSB
```


- FPC\$SENDG, SEND DATAGRAM

```
033F 1099      .SBTTL -      FPC$SENDG,      SEND DATAGRAM
033F 1100      .SBTTL -      FPC$SENDERG,      SEND DG, NO CDRP
033F 1101
033F 1102      :+
033F 1103      : FPC$SENDG formats and sends the caller-specified datagram. The
033F 1104      : SYSAP can specify via the flags input argument what happens to the
033F 1105      : buffer once it has been sent:
033F 1106      :
033F 1107      : flags = SYSAP$C_DISPQ implies that the buffer is placed
033F 1108      : on the datagram free queue for a future receive.
033F 1109      : The SYSAP's datagram receive count is incremented
033F 1110      : in the CDT in anticipation of the buffer going on
033F 1111      : the free queue.
033F 1112      :
033F 1113      : = SYSAP$C_DISPRET says that the SYSAP wants the sent
033F 1114      : buffer back, so RETFLAG is set to 1 (true) and
033F 1115      : DISPOSAL is 1.
033F 1116      :
033F 1117      : = SYSAP$C_DISPPQ says that the SYSAP wants SCS to
033F 1118      : put the sent buffer in nonpaged pool, so RETFLAG
033F 1119      : is set to 1 (true) and DISPOSAL = 0.
033F 1120      :
033F 1121      : This data is all expressed in table DG_SENT_FLGS.
033F 1122      :
033F 1123      : Inputs:
033F 1124      :
033F 1125      : R0 -Input flag described above
033F 1126      : R1 -Length of application data in dg
033F 1127      : R2 -Addr of dg, application data (FPC$SENDERG)
033F 1128      : R3 -Addr of CDT (FPC$SENDERG)
033F 1129      : R4 -Addr of PDT
033F 1130      : R5 -Addr of CDRP (FPC$SENDG)
033F 1131      : CDRP$L_CDT -Addr of CDT (FPC$SENDG)
033F 1132      : CDRP$L_MSG_BUF -Addr of datagram
033F 1133      :
033F 1134      : Outputs:
033F 1135      :
033F 1136      : R0 -Status: SS$_NORMAL, SS$_ILLCDTST
033F 1137      : R2 -Destroyed
033F 1138      : Other registers -Preserved
033F 1139      :
033F 1140      :
033F 1141      : .ENABL LSB
033F 1142      :
033F 1143      : FPC$SENDERG::
033F 1144      :
53 DD 033F 1145      : PUSH R3 ; Save caller's R3
0341 1146      : $CHK_CDTSTATE - ; Verify that connection state
0341 1147      : OPEN, - ; is open
0341 1148      : ERROR=STATE_ERR_R3,- ;
0341 1149      : CDT=R3 ;
16 11 034A 1150      : BRB 10$ ; Join common code
034C 1151      :
034C 1152      : FPC$SENDG::
034C 1153      :
53 24 53 DD 034C 1154      : PUSH R3 ; Save caller's R3
DO DO 034E 1155      : MOVL CDRP$L_CDT(R5),R3 ; Get addr of CDT
```


- FPC\$SENDRGDG, SEND DG, NO CDRP

			0352	1156	\$CHK_CDTSTATE -	: Verify that connection state		
			0352	1157	OPEN,-	: is open		
			0352	1158	ERROR=STATE_ERR_R3,-	: Else report error to SYSAP		
			0352	1159	CDT=R3	:		
52	1C	A5	D0	035B	1160	MOVL CDRP\$L_MSG_BUF(R5),R2	: Get addr of dg buff, appl data	
	1C	A5	D4	035F	1161	CLRL CDRP\$L_MSG_BUF(R5)	: Show dg is gone	
			0362	1162				
		50	95	0362	1163	10\$: TSTB R0	: Dg going on to free queue?	
		03	12	0364	1164	BNEQ 20\$: Branch if not	
	4C	A3	B6	0366	1165	INCW CDT\$W_DGREC(R3)	: Else step recv count in anticipation	
			0369	1166				
FO	A2	51	0E	A1	0369	1167	20\$: ADDW3 #SCS\$C_OVHD,R1,-	: Dg length = SCS header size +
					036E	1168	SCS\$W_LENGTH(R2)	: application data
			0B	3C	036E	1169	MOVZWL #SCS\$C_APPL_DG,-	: Set SCS type to application
	F4	A2			0370	1170	SCS\$W_MTYPE(R2)	: datagram
	14	A3	D0		0372	1171	MOVL CDT\$L_RCONID(R3),-	: Set destination connection
	F8	A2			0375	1172	SCS\$L_DST_CONID(R2)	: ID in SCS header
	18	A3	D0		0377	1173	MOVL CDT\$L_LCONID(R3),-	: Put local connection ID
	FC	A2			037A	1174	SCS\$L_SRC_CONID(R2)	: into header
51	1C	A3	D0		037C	1175	MOVL CDT\$L_PB(R3),R1	: Get address of PB in R1
	70	A3	D6		0380	1176	INCL CDT\$L_DGSENT(R3)	: Step count of application dgs sent
	FC	A3	30		0383	1177	BSBW INT\$SNDG	: Send datagram
		53	8ED0		0386	1178	POPL R3	: Restore caller's R3
	50	01	3C		0389	1179	MOVZWL #SS\$_NORMAL,R0	: Set status to success
			05		038C	1180	RSB	: Return to SYSAP
					038D	1181		
					038D	1182	.DSABL LSB	

BLOCK TRANSFER CALLS

```
038D 1184 .SBTTL BLOCK TRANSFER CALLS
038D 1185 .SBTTL - FPC$MAP, MAP A BUFFER
038D 1186 .SBTTL - FPC$MAPBYPASS, MAP A BUFFER W/
038D 1187 .SBTTL - NO ACCESS CHECKING
038D 1188 .SBTTL - FPC$MAPIRP, MAP A BUFFER W/
038D 1189 .SBTTL - ARGUMENTS IN IRP
038D 1190 .SBTTL - FPC$MAPIRPBYP, MAP A BUFFER W/
038D 1191 .SBTTL - ARGUMENTS IN IRP AND NO
038D 1192 .SBTTL - ACCESS CHECKING
038D 1193
038D 1194 :+
038D 1195 : Each of the entries converts its inputs to a set of common inputs:
038D 1196 :
038D 1197 : R1 -Addr of 3 longwd array containing
038D 1198 : SVAPTE, BOFF, and BCNT (size) of
038D 1199 : buffer to map.
038D 1200 : R2 -Buffer descriptor flags consisting of
038D 1201 : valid (bit 15), access mode = 0/1/2/3
038D 1202 : (bits 13,14), and access checking = 0/1
038D 1203 : for disabled/enabled (bit 12).
038D 1204 :
038D 1205 : Common map processing then consists of allocating a buffer descriptor
038D 1206 : from the pool (common to all CI ports), filling in the buffer descriptor
038D 1207 : and then filling in the SYSAP's buffer handle.
038D 1208 :
038D 1209 : If no buffer descriptor is available, then the common inputs are
038D 1210 : saved temporarily in the buffer handle provided by the SYSAP. The
038D 1211 : SCS MAP routine is suspended until resumed by the deallocation of a buffer
038D 1212 : descriptor. Upon resumption, all context is retrieved including R1
038D 1213 : and R2 and a buffer descriptor allocated.
038D 1214 :
038D 1215 : Inputs to all MAP calls:
038D 1216 :
038D 1217 : R4 -PDT addr
038D 1218 : R5 -CDRP addr
038D 1219 :
038D 1220 : CDRP$L_CDT -Addr of CDT
038D 1221 : CDRP$L_LBUFH_AD -Addr of SYSAP's buffer handle
038D 1222 :
038D 1223 : CDT$L_RCONID -Remote connection ID
038D 1224 :
038D 1225 : Inputs to MAP, MAPBYPASS:
038D 1226 :
038D 1227 : R1 -Addr of SVAPTE/BOFF/BCNT array
038D 1228 : R2 -Access mode = 0/1/2/3 for kernel/
038D 1229 : exec/super/user
038D 1230 :
038D 1231 : Inputs to MAPIRP, MAPIRPBYP:
038D 1232 :
038D 1233 : CDRP$L_SVAPTE(R5) = Addr of SVAPTE in IRP
038D 1234 : CDRP$B_RMOD(R5) = Addr of access mode
038D 1235 :
038D 1236 : Outputs for all map routines:
038D 1237 :
038D 1238 : @CDRP$L_LBUFH_AD(R5) -Filled in with byte offset of buffer,
038D 1239 : buffer name, local connection ID
038D 1240 :-
```


- ACCESS CHECKING

```
038D 1241
038D 1242      .ENABL  LSB
038D 1243
038D 1244 FPC$MAPIRBPBY::
038D 1245
51  CC A5 DE 038D 1246      MOVAL  CDRP$L_SVAPTE(R5),R1      ; Get addr in IRP of SVAPTE
52  AB A5 9A 0391 1247      MOVZBL CDRP$B_RMOD(R5),R2      ; and access mode
0395 1248
0395 1249 FPC$MAPBYPASS::
0395 1250
0395 1251      ASSUME  CIBD$V_V EQ 15
0395 1252
52  52 04 A8 0395 1253      BISW   #4,R2      ; Set valid bit to left of access mode
52  52 0D 78 0398 1254      ASHL   #CIBD$V_ACMOD,R2,R2      ; Position valid, access mode
19  11 039C 1255      BRB     MAP_COMMON      ; Join common code
039E 1256
039E 1257 FPC$MAPIRBP::
039E 1258
0044 8F B3 039E 1259      BITW   #<IRP$M_PAGIO!IRP$M_SWAPIO>,-
CA A5 03A2 1260      CDRP$W_STS(R5)      ; Is this page/swap I/O?
E7 12 03A4 1261      BNEQ    FPC$MAPIRBPBY      ; Branch if so to bypass
51  CC A5 DE 03A6 1262      MOVAL  CDRP$L_SVAPTE(R5),R1      ; Get addr in IRP of SVAPTE
52  AB A5 9A 03AA 1263      MOVZBL CDRP$B_RMOD(R5),R2      ; and access mode
03AE 1264
03AE 1265 FPC$MAP::
03AE 1266
52  52 0D 78 03AE 1267      ASHL   #CIBD$V_ACMOD,R2,R2      ; Position access mode
52  9000 8F A8 03B2 1268      BISW   #CIBD$M_V!CIBD$M_AC,R2      ; Set valid and access check
03B7 1269
03B7 1270 MAP_COMMON:
03B7 1271
18 A5 8ED0 03B7 1272      POPL   CDRP$L_SAVD_RTN(R5)      ; Pop return from stack to CDRP
03BB 1273
03BB 1274 ALLOC_BD:
03BB 1275
50  00000000'53 DD 03BB 1276      PUSHL  R3      ; Save SYSAP register
53  F4 A0 DO 03BD 1277      MOVL   G^SCS$GL_BDT,R0      ; Get addr of buffer desc table
44 13 03C4 1278      MOVL   CIBD$L_FREEBD(R0),R3      ; Get addr of 1st free desc
OC A3 DO 03C8 1279      BEQL   WAIT BD      ; Branch if none
F4 A0 03CA 1280      MOVL   CIBD$L_LINK(R3),-      ; Remove BD from linked
03CD 1281      CIBD$L_FREEBD(R0)      ; List
03CF 1282
03CF 1283      ASSUME  CDRP$L_SVAPTE+4 EQ CDRP$W_BOFF
03CF 1284      ASSUME  CDRP$W_BOFF+2 EQ CDRP$L_BCNT
03CF 1285
03CF 1286      ; Fill in buffer descriptor:
03CF 1287      MOVL   (R1)+,CIBD$L_SVAPTE(R3)      ; Addr of PTE mapping buff
63  81 52 A1 03D3 1288      ADDW3  R2,(R1)+,CIBD$W_FLAGS(R3)      ; Byte offset, access, valid
04 A3 61 DO 03D7 1289      MOVL   (R1),CIBD$L_BLEN(R3)      ; Size of buffer
OC A3 55 DO 03DB 1290      MOVL   R5,CIBD$L_CDRP(R3)      ; CDRP
03DF 1291
50  53 00000000'53 C3 03DF 1292      SUBL3  G^SCS$GL_BDT,R3,R0      ; Compute index
50  50 50 FC 8F 78 03E7 1293      ASHL   #-4,R0,R0      ; to buffer descriptor
50  10 10 02 A3 F0 03EC 1294      INSV   CIBD$W_KEY(R3),#16,#16,R0      ; Put seq # in h.o. bits
53  2C A5 DO 03F2 1295      ; to make buffer name
03F2 1296      MOVL   CDRP$L_LBUFH_AD(R5),R3      ; Get buffer handle to fill in
03F6 1297
```


- ACCESS CHECKING

				03F6	1298	ASSUME	CIBHANS\$_BOFF+4 EQ CIBHANS\$_BNAME	
				03F6	1299	ASSUME	CIBHANS\$_BNAME+4 EQ CIBHANS\$_RCONID	
				03F6	1300			
		83	D4	03F6	1301	CLRL	(R3)+	; Clear transfer offset
	83	50	D0	03F8	1302	MOVL	R0,(R3)+	; Copy buffer name
50	24	A5	D0	03FB	1303	MOVL	CDRPS\$_CDT(R5),R0	; Get CDT addr
63	14	A0	D0	03FF	1304	MOVL	CDT\$_RCONID(R0),(R3)	; Put CONID into handle
0094	C0	61	C0	0403	1305	ADDL	(R1),CDT\$_BYTMAPD(R0)	; Incr count of total bytes mapped
				0408	1306			; by the # bytes just mapped
		53	8ED0	0408	1307	POPL	R3	; Restore SYSAP's R3
	18	B5	17	040B	1308	JMP	@CDRPS\$_SAVD_RTN(R5)	; Return to SYSAP
				040E	1309			
				040E	1310			
				040E	1311			
		51	DD	040E	1312	PUSHL	R1	; Save SVAPTE arg temporarily
51	2C	A5	D0	0410	1313	MOVL	CDRPS\$_LBUFH_AD(R5),R1	; Get buffer handle addr
		61	8ED0	0414	1314	POPL	CIBHANS\$_BOFF(R1)	; Copy SVAPTE and access mode to
08	A1	52	D0	0417	1315	MOVL	R2,CIBHANS\$_RCONID(R1)	; handle for duration of suspend
	04	A1	D4	041B	1316	CLRL	CIBHANS\$_BNAME(R1)	; Zero buffer name to show
				041E	1317			; that none is allocated
		53	8ED0	041E	1318	POPL	R3	; Restore SYSAP's R3
	51	50	D0	0421	1319	MOVL	R0,R1	; Copy BDT addr to register not
				0424	1320			; used by \$SUSP_SCS macro
50	24	A5	D0	0424	1321	MOVL	CDRPS\$_CDT(R5),R0	; Get addr of CDT
009A	C0	B6		0428	1322	INCW	CDT\$_QBDT_CNT(R0)	; Incr count of # times suspended
				042C	1323			; waiting for BDT
				042C	1324			
				042C	1325			
51	2C	A5	D0	0445	1326	MOVL	@CIBDT\$_WAITBL(R1)	; Suspend this routine
52	08	A1	D0	0449	1327	MOVL	CDRPS\$_LBUFH_AD(R5),R1	; on availability of BD
	51	61	D0	044D	1328	MOVL	CIBHANS\$_RCONID(R1),R2	; Get addr of thread's buffer handle
	FF68	31		0450	1329	MOVL	CIBHANS\$_BOFF(R1),R1	; Retrieve access mode and SVAPTE
				0453	1330	BRW	ALLOC_BD	; saved over the suspend
				0453	1331			; Try to allocate now

.DSABL LSB

- FPC\$REQDATA, BLOCK XFER READ

```
0453 1333      .SBTTL -      FPC$REQDATA,      BLOCK XFER READ
0453 1334      .SBTTL -      FPC$SENDATA,      BLOCK XFER WRITE
0453 1335
0453 1336 :+
0453 1337 : These two calls are the same except for the direction of
0453 1338 : the block transfer. FPC$REQDATA runs as follows:
0453 1339 :
0453 1340 : 1. Using the CDT address specified in the SYSAP's remote buffer
0453 1341 : handle, fill in the allocated message buffer with the REQDAT
0453 1342 : opcode, remote station, and all frills set to 0. (512 byte
0453 1343 : data pkt, response bit off, path select auto.) The response
0453 1344 : bit = 0 will cause the REQDAT buffer to be put on the free
0453 1345 : queue once it has been sent where it will wait to receive the
0453 1346 : DATRET/DATREC notification of transfer completion.
0453 1347 :
0453 1348 : 2. Fill in the sender buffer name and byte offset with info
0453 1349 : from the remote buffer handle. Note that the net buffer offset
0453 1350 : is the sum of the offset in the buffer handle and the offset
0453 1351 : specified by the SYSAP in the CDRP. The buffer handle offset
0453 1352 : is normally 0. for third party transfers, it may be transformed
0453 1353 : by the SYSAP acting as the manager of the third party transaction
0453 1354 : in the case where that SYSAP discovers that it must break a
0453 1355 : transfer into transfers from different sources. The CDRP byte
0453 1356 : offset is intended for use by a SYSAP doing segmented transfers.
0453 1357 :
0453 1358 : 3. Fill in the receiver buffer name and byte offset with info
0453 1359 : from the local buffer handle.
0453 1360 :
0453 1361 : 4. Set the XCT_ID to the local CONID (from the local buffer handle)
0453 1362 : followed by the RSPID from the CDRP. Set the XCT_LEN to the
0453 1363 : value specified in the CDRP.
0453 1364 :
0453 1365 : 5. Map the RSPID to the CDRP, save the SYSAP's context in the CDRP;
0453 1366 : send the REQDAT message, and return to the caller's caller.
0453 1367 : The SYSAP remains suspended until the transfer completes at which
0453 1368 : time the SYSAP is resumed at the instruction following the call
0453 1369 : to request data.
0453 1370 :
0453 1371 : FPC$SENDATA has its own version of steps 1-3. In this case the
0453 1372 : send buffer information is in the local buffer handle and the receive
0453 1373 : buffer information is in the remote buffer handle.
0453 1374 :
0453 1375 : Inputs:
0453 1376 :
0453 1377 :      R4      -PDT addr
0453 1378 :      R5      -CDRP addr
0453 1379 :
0453 1380 :      CDRP$L_RSPID      -RSPID to use to correlate transfer
0453 1381 :                          completion with initiation thread
0453 1382 :      CDRP$L_MSG_BUF      -Message buffer to use for xfer command
0453 1383 :      CDRP$L_XCT_LEN      -# bytes to xfer
0453 1384 :      CDRP$L_LBUFH_AD      -Addr of local buffer handle
0453 1385 :      CDRP$L_LBOFF      -Local byte offset for segmentation
0453 1386 :      CDRP$L_RBUFH_AD      -Addr of remote buffer handle
0453 1387 :      CDRP$L_RBOFF      -Remote byte offset for segmentation
0453 1388 :
0453 1389 : Outputs:
```



```
0453 1390 :
0453 1391 :
0453 1392 :
0453 1393 :
0453 1394 :
0453 1395 :
0453 1396 :
0453 1397 :
0453 1398 :
0453 1399 :
0453 1400 :-
0453 1401 :
0453 1402 :
0453 1403 :
0453 1404 FPC$REQDATA::
0453 1405
51 53 DD 0453 1406 PUSH R3 ; Save SYSAP's R3
51 34 A5 D0 0455 1407 MOV CDRP$L_RBUFH AD(R5),R1 ; Get addr of remote buffer handle
53 08 A1 3C 0459 1408 MOVZWL CIBHANS$L_RCONID(R1),R3 ; Compute addr of CDT
50 00000000'GF D0 045D 1409 MOV G*SCS$GL_CDL,R0 ; specified by local
53 6043 D0 0464 1410 MOV (R0)[R3],R3 ; buffer handle
008C C3 D6 0468 1411 INCL CDT$L_REQDATS(R3) ; Incr number of request datas issued
3C A5 C0 046C 1412 ADDL CDRP$L_XCT_LEN(R5),- ; Step count of # bytes xferred via
0090 C3 046F 1413 CDT$L_BYTREQD(R3) ; all request datas
51 34 A5 D0 0472 1414 MOV CDRP$L_RBUFH AD(R5),R1 ; Get addr of remote buffer handle
52 1C A5 D0 0476 1415 MOV CDRP$L_MSG_BUF(R5),R2 ; Set pointer to SCS area
04 A1 D0 047A 1416 MOV CIBHANS$L_BNAME(R1),- ; Set send buffer name
FC A2 047D 1417 SCSS$L_SND_NAME(R2) ; to remote
61 C1 047F 1418 ADDL3 CIBHANS$L_BOFF(R1),- ; Set send byte offset to
38 A5 0481 1419 CDRP$L_RBOFF(R5),- ; xfer offset +
62 0483 1420 SCSS$L_SND_BOFF(R2) ; segmentation
51 2C A5 D0 0484 1421 MOV CDRP$L_LBUFH AD(R5),R1 ; Get local buffer handle
04 A1 D0 0488 1422 MOV CIBHANS$L_BNAME(R1),- ; Set receive buffer name
04 A2 048B 1423 SCSS$L_REC_NAME(R2) ; to local
61 C1 048D 1424 ADDL3 CIBHANS$L_BOFF(R1),- ; Set receive byte offset to
30 A5 048F 1425 CDRP$L_LBOFF(R5),- ; xfer offset
50 08 A2 0491 1426 SCSS$L_REC_BOFF(R2) ; + segmentation
0000'CF 9E 0493 1427 MOVAB W*INT$REQDAT,R0 ; Addr of PPD action routine
45 11 0498 1428 BRB COMMON_XFER ; Join common code
049A 1429
049A 1430 FPC$SENDATA::
049A 1431
51 53 DD 049A 1432 PUSH R3 ; Save SYSAP's R3
51 34 A5 D0 049C 1433 MOV CDRP$L_RBUFH AD(R5),R1 ; Get addr of remote buffer handle
53 08 A1 3C 04A0 1434 MOVZWL CIBHANS$L_RCONID(R1),R3 ; Compute addr of CDT
50 00000000'GF D0 04A4 1435 MOV G*SCS$GL_CDL,R0 ; specified by local
53 6043 D0 04AB 1436 MOV (R0)[R3],R3 ; buffer handle
0084 C3 D6 04AF 1437 INCL CDT$L_SND DATS(R3) ; Incr total # send datas issued
3C A5 C0 04B3 1438 ADDL CDRP$L_XCT_LEN(R5),- ; Step count of total bytes xferred via
0088 C3 04B6 1439 CDT$L_BYTSENT(R3) ; send datas
51 34 A5 D0 04B9 1440 MOV CDRP$L_RBUFH AD(R5),R1 ; Get addr of remote buffer handle
52 1C A5 D0 04BD 1441 MOV CDRP$L_MSG_BUF(R5),R2 ; Get base of buffer
04 A1 D0 04C1 1442 MOV CIBHANS$L_BNAME(R1),- ; Set receive buffer name
04 A2 04C4 1443 SCSS$L_REC_NAME(R2) ; to remote
61 C1 04C6 1444 ADDL3 CIBHANS$L_BOFF(R1),- ; Set receive byte offset to
38 A5 04C8 1445 CDRP$L_RBOFF(R5),- ; xfer offset +
08 A2 04CA 1446 SCSS$L_REC_BOFF(R2) ; segmentation
```


- FPC\$SENDATA, BLOCK XFER WRITE

```
51 2C A5 D0 04CC 1447      MOVL  CDRP$L_LBUFH_AD(R5),R1 ; Get local buffer handle
    04 A1 D0 04D0 1448      MOVL  CIBHAN$L_BNAME(R1),- ; Set send buffer name
    FC A2      04D3 1449      SCSSL_SND_NAME(R2) ; to local
    61 C1 04D5 1450      ADDL3  CIBHAN$L_BOFF(R1),- ; Set send byte offset to
    30 A5      04D7 1451      CDRP$L_LBOFF(R5),- ; xfer offset +
    62      04D9 1452      SCSSL_SND_BOFF(R2) ; segmentation
50 0000'CF 9E 04DA 1453      MOVAB W^INT$SNDDAT,R0 ; Addr of PPD action routine
    04DF 1454
    04DF 1455 COMMON_XFER:
    04DF 1456
    04DF 1457 $CHK_CDTSTATE - ; Verify connection state is
    04DF 1458 OPEN,- ; open.
    04DF 1459 ERROR=STATE_ERR_R3,- ; Else notify caller
    04DF 1460 CDT=R3
    18 A3 D0 04E8 1461      MOVL  CDT$L_LCONID(R3),- ; Set transaction ID =
    F0 A2      04EB 1462      SCSSL_LCONID(R2) ; local CONID followed
    20 A5 D0 04ED 1463      MOVL  CDRP$L_RSPID(R5),- ; by RSPID
    F4 A2      04F0 1464      SCSSL_RSPID(R2)
    3C A5 D0 04F2 1465      MOVL  CDRP$L_XCT_LEN(R5),- ; Set transfer size
    F8 A2      04F5 1466      SCSSL_XCT_LEN(R2)
51 1C A3 D0 04F7 1467      MOVL  CDT$L_PB(R3),R1 ; Get address of PB in R1
    60 16 04FB 1468      JSB (R0) ; Call the PPD layer
    1C A5 D4 04FD 1469      CLRL CDRP$L_MSG_BUF(R5) ; Zero msg buffer addr
    53 BED0 0500 1470      POPL R3 ; Restore SYSAP's R3
    0503 1471 $SUSP_FP ; Suspend caller
    050C 1472
    050C 1473 .DSABL LSB
```


- UNMAP, UNMAP A BUFFER

.SBTTL - UNMAP, UNMAP A BUFFER

050C 1475
050C 1476
050C 1477 :+
050C 1478 : UNMAP converts the buffer name specified in the local buffer handle
050C 1479 : to a buffer descriptor address. If the buffer descriptor is not
050C 1480 : good (sequence number check), then the routine bugchecks. Otherwise,
050C 1481 : the descriptor valid bit is cleared, the sequence number incremented,
050C 1482 : and the descriptor is linked to the free list. Any CDRP waiting for
050C 1483 : a buffer descriptor is resumed.
050C 1484

Inputs:

050C 1485
050C 1486
050C 1487 R4 -PDT addr
050C 1488 R5 -CDRP addr
050C 1489
050C 1490 CDRP\$L_LBUFH_AD -Addr of local buffer handle
050C 1491

Outputs:

050C 1492
050C 1493
050C 1494 R0-R2 -Destroyed
050C 1495 Other registers -Preserved
050C 1496
050C 1497 CIBHAN\$L_BNAME -Zeroed
050C 1498 :-
050C 1499

.ENABL LSB

FPC\$UNMAP::

50 51 2C A5 DO 050C 1504 MOVL CDRP\$L_LBUFH_AD(R5),R1 : Get addr of local buff handle
52 04 A1 DO 0510 1505 MOVL CIBHAN\$L_BNAME(R1),R2 : Get buffer name
49 13 0514 1506 BEQL 30\$: Branch if none allocated
52 52 3C 0516 1507 MOVZWL R2,R2 : Isolate BD index
00000000 GF DO 0519 1508 MOVL G^SCSS\$GL_BDT,R0 : Get addr of BDT
F8 A0 52 D1 0520 1509 CML R2,CIBDT\$L_MAXIDX(R0) : Index greater than maximum?
3A 14 0524 1510 BGTR BD_SEQ_ERROR : Branch if so, same as bad seq number
0526 1511
0526 1512 ASSUME CIBD\$C_LENGTH EQ 16
0526 1513
52 52 C0 0526 1514 ADDL R2,R2 : Prepare for net 16 byte index
52 6042 7E 0529 1515 MOVAQ (R0)[R2],R2 : Get addr of BD
02 A2 B1 052D 1516 CMPW CIBD\$W_KEY(R2),- : Sequence # in BD =
06 A1 0530 1517 CIBHAN\$L_BNAME+2(R1) : that in buffer handle?
2C 12 0532 1518 BNEQ BD_SEQ_ERROR : Branch if not
02 A2 B6 0534 1519 INCW CIBD\$W_KEY(R2) : Step sequence number
03 12 0537 1520 BNEQ 10\$: Branch if nonzero
02 A2 B6 0539 1521 INCW CIBD\$W_KEY(R2) : Else step again
053C 1522
0F E5 053C 1523 10\$: BBCC #CIBD\$V_V,- : Clear valid bit
00 62 053E 1524 CIBD\$W_FLAGS(R2),20\$:
0540 1525
F4 A0 DO 0540 1526 20\$: MOVL CIBDT\$L_FREEBD(R0),- : Link this BD to
OC A2 0543 1527 CIBD\$L_LINK(R2) : free list
F4 A0 52 DO 0545 1528 MOVL R2,CIBDT\$L_FREEBD(R0) :
04 A1 D4 0549 1529 CLRL CIBHAN\$L_BNAME(R1) : Zero buffer name to show
054C 1530 : none mapped
054C 1531 \$RESUME_FP - : Resume waiter, if nay

- UNMAP, UNMAP A BUFFER

```
054C 1532 @CIBDT$L_WAITFL(R0) ;
055F 1533 ;
05 055F 1534 30$: RSB ; Return to caller
0560 1535 ;
0560 1536 BD_SEQ_ERROR: ; SYSAP tried to unmap buffer
0560 1537 ;
0560 1538 BUGCHECK CIPORT, NONFATAL ; without right key -- leave
0567 1539 ; buffer descriptor permanently
0567 1540 ; allocated and do nothing to it.
05 0567 1541 40$: RSB ; return to caller
0568 1542 ;
0568 1543 ;
0568 1544 .DSABL LSB
```


- SUSP_CONCALL, SUSPEND CONNECTION

```
0568 1546      .SBTTL -      SUSP_CONCALL,  SUSPEND CONNECTION
0568 1547      .SBTTL -      MANAGEMENT CALL
0568 1548
0568 1549 :+
0568 1550 : Connection management calls assume that the SYSAP's fork process
0568 1551 : consists of R3 = CDT address, R4 = PDT address, R5, and (SP) =
0568 1552 : return from the connection management call. R3 is automatically
0568 1553 : restored by the event (response) triggering call completion; R4
0568 1554 : is restorable from the CDT. Therefore, the only context saved is
0568 1555 : R5 and return from call.
0568 1556
0568 1557 : Inputs:
0568 1558
0568 1559 :      R3      -CDT addr
0568 1560 :      R4      -PDT addr
0568 1561 :      R5      -SYSAP's R5
0568 1562 :      (SP)    -SYSAP PC
0568 1563
0568 1564 : Outputs:
0568 1565
0568 1566 :      R5, (SP)+ -Saved in CDT
0568 1567 :      Return to caller's caller
0568 1568 :-
0568 1569
0568 1570      .ENABL  LSB
0568 1571
0568 1572 SUSP_CONCALL:
0568 1573
0568 1574      MOVL    R5,CDT$$_FR5(R3)      ; Save SYSAP R5
0568 1575      POPL    CDT$$_FPC(R3)      ; Save SYSAP PC and remove it from stack
0568 1576      RSB
0568 1577      ; Return to caller's caller
0571 1577
0571 1578      .DSABL  LSB
```

68 A3 55 D0
64 A3 8ED0
05

- STATE_ERR, RETURN CDT STATE ERROR

```
0571 1580      .SBTTL -      STATE_ERR,      RETURN CDT STATE ERROR
0571 1581      .SBTTL -      TO SYSAP
0571 1582
0571 1583      ;+
0571 1584      ; Set error status code and return to caller.
0571 1585      ; -
0571 1586
0571 1587 STATE_ERR_R3:      ; Entry if caller's R3 is saved on stack
0571 1588
53 8ED0 0571 1589      POPL      R3      ; Restore R3 for caller
0574 1590
0574 1591 STATE_ERR:
0574 1592
50 2154 8F 3C 0574 1593      MOVZWL #SS$_ILLCDTST,R0      ; Status = illegal CDT state
05 05 0579 1594      RSB      ; Return to SYSAP
```


MAINTENANCE FUNCTION CALLS

```
057A 1596      .SBTTL MAINTENANCE FUNCTION CALLS
057A 1597      .SBTTL -      FPC$READCOUNT,  READ AND LOCK
057A 1598      .SBTTL -      PORT COUNTERS
057A 1599
057A 1600 :+
057A 1601 : This routine is called by a SYSAP to reset the port counters to begin
057A 1602 : counting ACKS/NAKS/NO RESPONSES on each path and total datagrams discarded
057A 1603 : from a particular port or all ports. The SYSAP 'owns' the counters until it
057A 1604 : does a RLS_COUNTERS call. If another SYSAP owns the counters, then
057A 1605 : error status is returned to the SYSAP.
057A 1606
057A 1607 : Note that this is an unusual fork process call in that the SYSAP hands
057A 1608 : FPC$READCOUNT the base of the PPD layer of the dg pkt, and receives
057A 1609 : back the PPD layer address of the counters read response. The reason
057A 1610 : is that in this one case the application data is entirely port specific.
057A 1611 : The mechanism for managing counter ownership is all that is assumed to
057A 1612 : be port independent and hence can be handled in this module (which must
057A 1613 : be port independent.) The packet address is simply passed through this
057A 1614 : layer to the PPD layer without being used in any way. Future port
057A 1615 : implementations may have different counter management and, in that
057A 1616 : case counter ownership book keeping would also have to migrate into
057A 1617 : the PPD layer.
057A 1618
057A 1619 : Inputs:
057A 1620
057A 1621 :      R0      -Addr of remote station to count for;
057A 1622 :              0 addr means count for all stations
057A 1623 :      R1      -Addr of local process name
057A 1624 :      R2      -Addr of base of datagram sized buffer
057A 1625 :              (PPD layer)
057A 1626 :      R4      -Addr of PDT
057A 1627 :      R5      -Addr of CDRP
057A 1628
057A 1629 : Outputs:
057A 1630
057A 1631 :      R0      -Status: SSS_NORMAL, SSS_INTERLOCK,
057A 1632 :              SSS_NOSUCHNODE
057A 1633 :      R2      -Addr of datagram buffer, current counters
057A 1634 :              to all ports since last release
057A 1635 :      R1      -Destroyed
057A 1636
057A 1637 :      Other registers      -Preserved
057A 1638
057A 1639 :      PDT$B_FLAGS(R4)      -Counters busy flag set
057A 1640 :      PDT$T_CNTOWNER(R4)   -Name of owning SYSAP
057A 1641
057A 1642 :      PPD$L_PO_ACK(R2)     -ACKS on path 0
057A 1643 :      PPD$L_PO_NAK(R2)     -NAKS on path 0
057A 1644 :      PPD$L_PO_NRSP(R2)    -No responses on path 0
057A 1645 :      PPD$L_P1_ACK(R2)     -ACKS on path 1
057A 1646 :      PPD$L_P1_NAK(R2)     -NAKS on path 1
057A 1647 :      PPD$L_P1_NRSP(R2)    -No responses on path 1
057A 1648 :      PPD$L_DG_DISC(R2)    -Datagrams discarded
057A 1649 :-
057A 1650
057A 1651      .ENABL  LSB
057A 1652
```



```
- PORT COUNTERS

057A 1653 FPC$READCOUNT::
057A 1654
057A 1655 BBSS #PDT$V_CNTBSY,- ; Branch if counters busy; else
057C 1656 PDTSW_FLAGS(R4),BSY_ERR ; set busy and continue
23 00C0 00 E2 0580 1657 MOVQ (R1)+,PDT$T_CNTOWNER(R4); Save new owner's name
00C4 C4 81 7D 0585 1658 MOVQ (R1),PDT$T_CNTOWNER+8(R4)
00CC C4 61 7D 058A 1659
058A 1660 10$: BICW #PDT$M_CNTRL$,- ; Clear release pending
00C0 C4 02 AA 058A 1661 PDTSW_FLAGS(R4) ;
058C 1662
058F 1663 ISSUE_RDCNT:
058F 1664
058F 1665 BSBW INT$READCNT ; Issue command to port
FA6E' 30 058F 1666 BLBC R0,30$ ; If error, leave now
00D4 C4 25 50 E9 0592 1666 MOVQ R5,PDT$T_CNTCDRP(R4) ; Save caller's CDRP addr
00D4 C4 55 D0 0595 1667 $SUSP_FP ; Save fork process' context
059A 1668 ; till response arrives
05A3 1669
05A3 1670
05A3 1671 BSY_ERR:
05A3 1672
05A3 1673 PUSHR #^M<R0,R2,R3> ; Save registers for CMPC
61 0D BB 05A3 1674 CMPC3 #16,(R1),- ; Is current owner = requestor?
00C4 C4 29 05A5 1674 PDTSW_FLAGS(R4) ;
05A8 1675 ;
05AB 1676 TSTL R0 ; Check compare result
04 12 05AD 1677 BNEQ 20$ ; Branch if requestor not owner
0D BA 05AF 1678 POPR #^M<R0,R2,R3> ; Restore registers
D7 11 05B1 1679 BRB 10$ ; Continue with request
05B3 1680
05B3 1681 20$: POPR #^M<R0,R2,R3> ; Restore registers
50 038C 0D BA 05B3 1681 20$: POPR #^M<R0,R2,R3> ; Restore registers
038C 8F 3C 05B5 1682 MOVZWL #SS$_INTERLOCK,R0 ; Else set error status
05 05BA 1683 30$: RSB ; Return to SYSAP
05BB 1684
05BB 1685 .DSABL LSB
```


- FPC\$RLSCOUNT, READ AND RELEASE

FPC\$RLSCOUNT, READ AND RELEASE
PORT COUNTERS

```
05BB 1687 .SBTTL -
05BB 1688 .SBTTL -
05BB 1689
05BB 1690 ;+
05BB 1691 ; FPC$RLSCOUNT has the same function as FPC$READCOUNT except that the
05BB 1692 ; caller is assumed to already own the counters so no check is done, and
05BB 1693 ; the port is reset to count all ports again. (Count all is the default
05BB 1694 ; while the counters are unowned.)
05BB 1695
05BB 1696 ; Inputs:
05BB 1697 ;
05BB 1698 ; R2 -Addr of base of dg sized buffer
05BB 1699 ; R4 -Addr of PDT
05BB 1700 ; R5 -Addr of CDRP
05BB 1701
05BB 1702 ; Outputs:
05BB 1703 ;
05BB 1704 ; R0 -Status: $$$_NORMAL
05BB 1705 ; R2 -Addr of datagram buffer filled
05BB 1706 ; as specified in FPC$READCOUNT
05BB 1707 ; R1 -Destroyed
05BB 1708
05BB 1709 ; Other registers -Preserved
05BB 1710 ; -
05BB 1711
05BB 1712 .ENABL LSB
05BB 1713
05BB 1714 FPC$RLSCOUNT::
05BB 1715
00C0 02 A8 05BB 1716 BISW #PDT$M_CNTRL$,- ; Set count release pending
C4 05BD 1717 PDT$W_FLAGS(R4) ;
50 D4 05C0 1718 CLRL R0 ; Set port to count all ports
CB 11 05C2 1719 BRB ISSUE_RDCNT ; Go give read count command
05C4 1720
05C4 1721 .DSABL LSB
```


- FPC\$MRESET, RESET REMOTE PORT/SYSTEM

```
05C4 1723 .SBTTL - FPC$MRESET, RESET REMOTE PORT/SYSTEM
05C4 1724
05C4 1725 :+
05C4 1726 : FPC$MRESET allocates a datagram buffer and uses it to send
05C4 1727 : a maintenacne reset to the specified remote port.
05C4 1728 :
05C4 1729 : Inputs:
05C4 1730 :
05C4 1731 : R0 -0/1 for dont/do force reset
05C4 1732 : R1 -Addr of remote station to reset
05C4 1733 : R4 -Addr of PDT
05C4 1734 :
05C4 1735 : Outputs:
05C4 1736 :
05C4 1737 : R0 -Status: SSS NORMAL, SSS_INSFMEM,
05C4 1738 : SSS NOSUCHNODE
05C4 1739 : R1,R2 -Destroyed
05C4 1740 :
05C4 1741 : Other registers -Preserved
05C4 1742 :-
05C4 1743
05C4 1744 .ENABL LSB
05C4 1745
05C4 1746 FPC$MRESET::
05C4 1747
0000 53 DD 05C4 1748 PUSHL R3 ; Save SYSAP register
CF 9F 05C6 1749 PUSHAB W^INT$MRESET ; PPD action routine
06 11 05CA 1750 BRB 10$ ; Join commond code
```


- FPC\$MSTART, SEND START TO REMOTE

```
05CC 1752      .SBTTL - FPC$MSTART, SEND START TO REMOTE
05CC 1753      .SBTTL - SYSTEM
05CC 1754
05CC 1755      ;+
05CC 1756      ; FPC$MSTART allocates a datagram buffer and sends a start command
05CC 1757      ; to the specified remote port/system.
05CC 1758
05CC 1759      ; Inputs:
05CC 1760      ;
05CC 1761      ; R0 -1/0 for use default start addr/
05CC 1762      ; specified start addr
05CC 1763      ; R1 -Addr of remote station addr
05CC 1764      ; R2 -Start addr to send if R0 = 0
05CC 1765      ; R4 -Addr of PDT
05CC 1766
05CC 1767      ; Outputs:
05CC 1768      ;
05CC 1769      ; R0 -Status: SSS NORMAL, SSS_INSFMEM,
05CC 1770      ; SSS_NOSUCHNODE
05CC 1771      ; R1,R2 -Destroyed
05CC 1772      ;
05CC 1773      ; Other registers -Preserved
05CC 1774      ;-
05CC 1775
05CC 1776
05CC 1777 FPC$MSTART::
05CC 1778
05CC 1779      PUSHL R3 ; Save SYSAP register
0000'CF 07 DD 05CE 1780      PUSHAB W^INT$MSTART ; PPD action routine
05D2 1781
05D2 1782 10$: PUSHHR #^M<R0,R1,R2> ; Save input arguments
07 BB 05D4 1783      BSBW INT$ALLOC_DG ; Get a dg buffer
FA29' 30 05D7 1784      BLBC R0, MEM_ERR ; Branch if none
0B 50 E9 05DA 1785      POPR #^M<R0,R1,R3> ; Retrieve two input arguments
0B BA 05DC 1786      JSB @ (SP)+ ; Issue command
9E 16 05DE 1787      POPL R3 ; Restore register
53 8ED0 05E1 1788      BLBC R0, PORT_ERR ; Bad port status
OE 50 E9 05E4 1789      RSB ; Return to SYSAP
05 05E5 1790
05E5 1791 MEM_ERR:
05E5 1792
05E5 1793      POPR #^M<R0,R1,R2> ; Clear input arguments
07 BA 05E7 1794      TSTL (SP)+ ; Clear PPD routine address
8E D5 05E9 1795      POPL R3 ; Restore SYSAP's R3
53 8ED0 05EC 1796      MOVZWL #SS$_INSFMEM, R0 ; Set error status
50 0124 8F 05F1 1797      RSB ; and return to SYSAP
05 05F2 1798
05F2 1799 PORT_ERR:
05F2 1800
05F2 1801      PUSHL R0 ; Save status
FA09' 30 05F4 1802      BSBW INT$DEAL_DG ; Get rid of the buffer
50 8ED0 05F7 1803      POPL R0 ; Restore status
05 05FA 1804      RSB
05FB 1805
05FB 1806      .DSABL LSB
```



```
- FPC$STOP_VCS, SEND SHUTDOWN ON ALL VCS

05FB 1808      .SBTTL -      FPC$STOP_VCS,  SEND SHUTDOWN ON ALL VCS
05FB 1809
05FB 1810      ;+
05FB 1811      ; FPC$STOP_VCS is very port specific. All we do here is call the port
05FB 1812      ; dependent routine, CNF$STOP_VCS which attempts to send a datagram
05FB 1813      ; to each known port. The datagram notifies the remote system that the
05FB 1814      ; host is shutting down, so it can notify its SYSAPs promptly of the event.
05FB 1815      ;
05FB 1816      ; Inputs:
05FB 1817      ;
05FB 1818      ;      R4                      -PDT address
05FB 1819      ;
05FB 1820      ; Outputs:
05FB 1821      ;
05FB 1822      ;      R0-R3                  -Destroyed
05FB 1823      ;
05FB 1824      ;      Other registers      -Preserved
05FB 1825      ;-
05FB 1826
05FB 1827      .ENABL  LSB
05FB 1828
05FB 1829 FPC$STOP_VCS::
05FB 1830
FA02' 30 05FB 1831      BSBW  CNF$STOP_VCS      ; Call routine that executes function
05FE 1832      RSB
05FF 1833
05FF 1834      .DSABL  LSB
```


RECEIVED PACKET ROUTINES

```
05FF 1836 .SBTTL RECEIVED PACKET ROUTINES
05FF 1837 .SBTTL - FPC$REC_DGREC, PROCESS RECEIVED DG
05FF 1838
05FF 1839 :+
05FF 1840 : FPC$REC_DGREC verifies the destination connection ID and checks that
05FF 1841 : the connection has at least one datagram queued for receive. If the
05FF 1842 : connection has no datagrams queued for receive, then the datagram is
05FF 1843 : discarded to the free queue and not given to the SYSAP. Otherwise,
05FF 1844 : the SYSAP's datagram input address is called. Upon return from the
05FF 1845 : SYSAP, control is returned to the INTR module to get the next response.
05FF 1846 :
05FF 1847 : Inputs:
05FF 1848 :
05FF 1849 : R2 -Addr of message buffer (user portion)
05FF 1850 : R4 -Addr of PDT
05FF 1851 :
05FF 1852 : Outputs:
05FF 1853 :
05FF 1854 : R4 -Preserved
05FF 1855 : Other registers -Destroyed
05FF 1856 :
05FF 1857 :-
05FF 1858
05FF 1859 ASSUME SYSAP$C_DGREC EQ 0
05FF 1860
05FF 1861 .ENABL LSB
05FF 1862
05FF 1863 FPC$REC_DGREC::
05FF 1864
013D 30 05FF 1865 BSBW FPC$CHK_DCONID : Verify destination CONID in
0602 1866 : SCS header
10 50 E9 0602 1867 BLBC R0,20$ : Branch if bad CONID
50 D4 0605 1868 CLRL R0 : Set flag to show DGREC
4C A3 B7 0607 1869 DECW CDT$W_DGREC(R3) : Decrement DG receive count
0A 18 060A 1870 BGEQ 30$ : Branch if rcv dg's available
4C A3 B6 060C 1871 INCW CDT$W_DGREC(R3) : Restore correct count
F9EE' 30 060F 1872 BSBW INT$INS_DFREQ : Get rid of dg
78 A3 D6 0612 1873 INCL CDT$W_DGDISCARD(R3) : Step dg discard count
05 0615 1874 20$: RSB
0616 1875
74 A3 D6 0616 1876 30$: INCL CDT$W_DGRCVD(R3) : Step count of total bytes of
0619 1877 : application data received
09 11 0619 1878 BRB DGCOM : Join common code
061B 1879
061B 1880 .DSABL LSB
```


- FPC\$REC_SNDDG, PROCESS SENT DG

.SBTTL - FPC\$REC_SNDDG, PROCESS SENT DG

061B 1882
061B 1883
061B 1884 :+
061B 1885 : FPC\$REC_SNDDG verifies the source connection ID. If correct, R0 is
061B 1886 : set to SYSAP\$C_DGSNT to indicate that the datagram is a sent DG
061B 1887 : rather than a new received DG. The correct length is set in R1.
061B 1888 :
061B 1889 : Inputs:
061B 1890 :
061B 1891 : R2 -Addr of dg buffer (user portion)
061B 1892 : R4 -Addr of PDT
061B 1893 :
061B 1894 : Outputs:
061B 1895 :
061B 1896 : R4 -Preserved
061B 1897 : Other registers -Destroyed
061B 1898 :
061B 1899 :-

.ENABL LSB

FPC\$REC_SNDDG::

00FB	30	061B	1905	BSBW	FPC\$CHK_SCONID	: Verify sending connection ID	
14 50	E9	061E	1906	BLBC	R0,10\$: Branch if invalid	
50 01	9A	0621	1907	MOVZBL	#SYSAP\$C_DGSNT,R0	: Set flag to indicate DGSNT	
		0624	1908				
		0624	1909	DGCOM:			
		0624	1910				
51	OE	A3	0624	SUBW3	#SCSSC_OVHD,-	: Application data = DG length -	
	F0	A2	0626		SCSSW_LENGTH(R2),R1	: SCS header size	
51	51	3C	0629	MOVZWL	R1,R1	: Expand to longword	
	54	DD	062C	PUSHL	R4	: Save R4(PDT) for REM_NEXT_RSP	
04	B3	16	062E	JSB	@CDT\$C_DGINPUT(R3)	: Call SYSAP to dispose of dg buffer	
	54	8ED0	0631	POPL	R4	: Restore	
		05	0634	RSB		: Return	
			0635				
F9C8'	31		0635	1919	10\$: BRW	INT\$INS_DFREQ	: Return dg to free queue and
			0638	1920			: RSB
			0638	1921	.DSABL	LSB	


```
0638 1923      .SBTTL -      FPC$REC_DATREC, PROCESS RECEIVED RETDAT
0638 1924      .SBTTL -      FPC$REC_CNFREC, PROCESS RECEIVED RETCNF
0638 1925
0638 1926      ;+
0638 1927      ; These routines perform the same steps.
0638 1928
0638 1929      ; First, the CONID portion of the XCT_ID is verified and converted to
0638 1930      ; a CDT address. The RSPID portion of the XCT_ID is converted to the
0638 1931      ; response descriptor address and the CDRP address extracted from the
0638 1932      ; RD. The RSPID and message buffer containing the CNFREC/DATREC are
0638 1933      ; then deallocated. Finally, the context of the suspended SYSAP is
0638 1934      ; restored and the SYSAP called back at the PC following the call to
0638 1935      ; send/request data.
0638 1936
0638 1937      Inputs:
0638 1938
0638 1939      R2      -Addr of message buffer (user portion)
0638 1940      R4      -Addr of PDT
0638 1941
0638 1942      Outputs:
0638 1943
0638 1944      R4      -Preserved
0638 1945      Other registers -Destroyed
0638 1946
0638 1947      :-
0638 1948
0638 1949      .ENABL  LSB
0638 1950
0638 1951      FPC$REC_DATREC::
0638 1952
0638 1953      FPC$REC_CNFREC::
0638 1954
0638 1955      BSBW      FPC$CHK_LCONID      ; Verify transaction ID/CONID
0638 1956
0638 1957      BLBC      R0,STALE CDT      ; and get CDT addr
0638 1958      INCW      CDT$W_SEND(R3)    ; Branch if stale CDT
0638 1959      MOVZWL     SC$S_L_RSPID(R2),R1 ; Add implied credit of 1
0638 1960      MOVL      G^SC$S_GL_RDT,R0   ; Get RSPID index
0638 1961      MOVAQ      (R0)[R1],R1        ; Get base of RD table
0638 1962      CMPW      RDSW_SEQNUM(R1),-   ; Get RD address
0638 1963      SC$S_C_RSPID+2(R2)            ; Verify
0638 1964      BNEQ      RD_SEQ_ERR          ; sequence number
0638 1965      MOVL      RDSL_CDRP(R1),R5    ; Branch if bad sequence number
0638 1966      PUSHL     R2                 ; Get CDRP addr
0638 1967      DEALLOC   RSPID             ; Save volatile register
0638 1968      POPL      R2                 ; Deallocate RSPID
0638 1969      BSBW      INT$DEAL_MSG        ; Restore register
0638 1970
0638 1971      ; Deallocate msg buffer to
0638 1972      ; pool since it is always allocated
0638 1973      ; from pool.
0638 1974      MOVQ      R3,-(SP)            ; Save CDT & PDT addr
0638 1975      MOVQ      CDRP$L_FR3(R5),R3   ; Restore SYSAP's R3,R4
0638 1976      MOVZWL     #SS$_NORMAL,R0     ; Set status to success
0638 1977      JSB        @CDRP$L_FPC(R5)    ; Call SYSAP back
0638 1978      MOVQ      (SP)+,R3            ; Restore CDT & PDT addr
0638 1979      BRB        CHK_CRWAIT         ; Join common code in REC_MSGREC
0638 1980
0638 1981      ; to start anyone waiting for
0638 1982      ; send credit, then go for next
```

50	00000000	'GF	D0	0641	1959	MOVZWL	SC\$S_L_RSPID(R2),R1	
51	6041	7E	064C	1961	MOVAQ	(R0)[R1],R1		
	06 A1	B1	0650	1962	CMPW	RDSW_SEQNUM(R1),-		
	F6 A2		0653	1963	SC\$S_C_RSPID+2(R2)			
	23	12	0655	1964	BNEQ	RD_SEQ_ERR		
55	61	D0	0657	1965	MOVL	RDSL_CDRP(R1),R5		
	52	DD	065A	1966	PUSHL	R2		
	52 8ED0		065C	1967	DEALLOC	RSPID		
	F998'	30	0662	1968	POPL	R2		
			0665	1969	BSBW	INT\$DEAL_MSG		
			0668	1970				
			0668	1971				
53	7E	53	7D	0668	1972	MOVQ	R3,-(SP)	
	10 A5	7D	066B	1973	MOVQ	CDRP\$L_FR3(R5),R3		
	50 01	3C	066F	1974	MOVZWL	#SS\$_NORMAL,R0		
	OC B5	16	0672	1975	JSB	@CDRP\$L_FPC(R5)		
53	8E	7D	0675	1976	MOVQ	(SP)+,R3		
	51	11	0678	1977	BRB	CHK_CRWAIT		
			067A	1978				
			067A	1979				


```
067A 1980 ; response
067A 1981 RD_SEQ_ERR:
067A 1982
067A 1983 $DEBUGCHECK #ERR$V DEB XCTER ; Optionally, bugcheck on this error
52 00B4 C4 C2 068D 1984 SUBL PDT$L MSGHDRSZ(R4),R2 ; Back up msg pointer to start of buffer
    F96B' 30 0692 1985 BSBW CNF$LRP PB MSG ; Given msg, look up PB if any
    F968' 31 0695 1986 BRW INT$CRASH_PORT ; Crash the port & restart
    0698 1987
    0698 1988 STALE_CDT:
    05 0698 1989 RSB ; All cleaned up, just return
    0699 1991
    0699 1992 .DSABL LSB
```


- FPC\$REC_MSGREC, PROCESS RECEIVED MSG

```
0699 1994 .SBTTL - FPC$REC_MSGREC, PROCESS RECEIVED MSG
0699 1995
0699 1996
0699 1997 :+ FPC$REC_MSGREC checks the SCS message type field. If the type code
0699 1998 : is SCS$C_APPL_MSG, then processing continues. Otherwise the message
0699 1999 : is an SCS control message and routine SCS$REC_SCSMSG is called.
0699 2000
0699 2001 : For application messages, it checks that the destination connection
0699 2002 : ID is legal. If not, the message buffer is discarded (returned to
0699 2003 : the free queue) and processing ends. Otherwise, the connection credit
0699 2004 : bookkeeping is done and the SYSAP's message input address is called.
0699 2005 : The SYSAP is responsible for disposing of the message buffer. Upon
0699 2006 : return from the SYSAP, REC_MSGREC branches to REM_NEXT_RSP.
0699 2007
0699 2008 : Inputs:
0699 2009
0699 2010 : R2 -Addr of message buffer (user portion)
0699 2011 : R4 -Addr of PDT
0699 2012
0699 2013 : Outputs:
0699 2014
0699 2015 : R4 -Preserved
0699 2016 : Other registers -Destroyed
0699 2017
0699 2018 :-
0699 2019
0699 2020 .ENABL LSB
0699 2021
0699 2022 FPC$REC_MSGREC::
0699 2023
0699 2024 F4 A2 B1 0699 2024 CMPW SCS$W_MTYPE(R2),- : Is this an application
0699 2025 0A 069C 2025 #SCS$C_APPL_MSG : message?
0699 2026 09 13 069D 2026 BEQL 10$ : Branch if yes
0699 2027 54 DD 069F 2027 PUSHL R4 : Save R4(PDT) for REM_NEXT_RSP
0699 2028 F95C' 30 06A1 2028 BSBW SCS$REC_SCSMSG : Message is SCS control- go handle
0699 2029 54 8ED0 06A4 2029 POPL R4 : Restore
0699 2030 05 06A7 2030 RSB : Get next response
0699 2031 06A8 2031
0699 2032 0094 30 06A8 2032 10$: BSBW FPC$CHK_DCONID : Verify destination CONID
0699 2033 37 50 E9 06AB 2033 BLBC R0,20$ : Branch if invalid
0699 2034 42 A3 B7 06AE 2034 DECW CDT$W_REC(R3) : Decrement send credit held
0699 2035 06B1 2035 : by remote
0699 2036 F6 A2 A0 06B1 2036 ADDW SCS$W_CREDIT(R2),- : Add credit extended by remote to
0699 2037 40 A3 06B4 2037 CDT$W_SEND(R3) : to send credit
0699 2038 0080 C3 D6 06B6 2038 INCL CDT$L_MSGRCVD(R3) : Incr count of # appl msgs received
0699 2039 0E A3 06BA 2039 SUBW3 #SCS$C_OVHD,-
0699 2040 51 F0 A2 06BC 2040 SCS$W_LENGTH(R2),R1 : Set size of application data
0699 2041 51 51 3C 06BF 2041 MOVZWL R1,R1 : for SYSAP
0699 2042 7E 53 7D 06C2 2042 MOVQ R3,-(SP) : Save CDT & PDT address
0699 2043 00 B3 16 06C5 2043 JSB @CDT$L_MSGINPUT(R3) : Call SYSAP message input address
0699 2044 53 8E 7D 06C8 2044 MOVQ (SP)+,R3 : Retrieve CDT & PDT address
0699 2045 06CB 2045
0699 2046 06CB 2046 CHK_CRWAIT:
0699 2047 06CB 2047
0699 2048 40 A3 B5 06CB 2048 TSTW CDT$W_SEND(R3) : Any send credit?
0699 2049 15 13 06CE 2049 BEQL 20$ : Branch if not
0699 2050 06D0 2050 $RESUME_FP - : Else, resume next waiter.
```


- FPC\$REC_MSGREC, PROCESS RECEIVED MSG

		06D0	2051		@CDT\$L_CRWAITQFL(R3),-	:
		06D0	2052		QEMPTY=20\$:
E6	11	06E3	2053	BRB	CHK_CRWAIT	: branching if none.
		06E5	2054			: Check for more credit
	05	06E5	2055	20\$:	RSB	
		06E6	2056			
		06E6	2057	.DSABL	LSB	

- FPC\$REC_SNDMSG, PROCESS SEND MSG

```
06E6 2059      .SBTTL -      FPC$REC_SNDMSG, PROCESS SEND MSG
06E6 2060
06E6 2061      :+
06E6 2062      : FPC$REC_SNDMSG simply calls FPC$DEALRGMSG to deallocate the sent
06E6 2063      : message. The deallocate takes care of flow control and may
06E6 2064      : deallocate the buffer to the free queue if the free queue is
06E6 2065      : low, or to pool.
06E6 2066      :
06E6 2067      : Inputs:
06E6 2068      :
06E6 2069      :      R2      -Addr of message buffer (user portion)
06E6 2070      :      R4      -Addr of PDT
06E6 2071      :
06E6 2072      : Outputs:
06E6 2073      :
06E6 2074      :      R4      -Preserved
06E6 2075      :      Other registers -Destroyed
06E6 2076      :
06E6 2077      :-
06E6 2078
06E6 2079      .ENABL  LSB
06E6 2080
06E6 2081 FPC$REC_SNDMSG::
06E6 2082
06E6 2083      BSBB  FPC$CHK_SCONID      : Verify source
06E6 2084      BLBC  R0,SC_SEQ_ERR    : connect ID
06E6 2085      BRW   FPC$DEALRGMSG   : Deallocate buffer
06E6 2086
06E6 2087 SC_SEQ_ERR:
06E6 2088
06E6 2089      SUBL  PDT$L_MSGHDRSZ(R4),R2 : Back up message addr to top
06E6 2090      : of buffer from user data
06E6 2091      BSBW  CNF$LKP_PB_MSG        : Given msg, look up PB, if any
06E6 2092      BRW   INT$CRASH_PORT
06E6 2093
06E6 2094      .DSABL  LSB
```

31 10 06E6 2083 BSBB FPC\$CHK_SCONID : Verify source
03 50 E9 06E6 2084 BLBC R0,SC_SEQ_ERR : connect ID
FB3D 31 06E6 2085 BRW FPC\$DEALRGMSG : Deallocate buffer
06EE 2086
06EE 2087 SC_SEQ_ERR:
06EE 2088
52 00B4 C4 C2 06EE 2089 SUBL PDT\$L_MSGHDRSZ(R4),R2 : Back up message addr to top
F90A' 30 06E6 2091 BSBW CNF\$LKP_PB_MSG : Given msg, look up PB, if any
F907' 31 06E6 2092 BRW INT\$CRASH_PORT
06F9 2093
06F9 2094 .DSABL LSB

- FPC\$REC_RDCNT, PROCESS RECEIVED RDCNT

```
06F9 2096 .SBTTL - FPC$REC_RDCNT, PROCESS RECEIVED RDCNT
06F9 2097
06F9 2098
06F9 2099 :+ FPC$REC_RDCNT returns the received buffer of port counters to the
06F9 2100 : SYSAP that owns the port counters currently. If the SYSAP specified
06F9 2101 : a release of the counters, then the counters busy flag is cleared.
06F9 2102 :
06F9 2103 : Inputs:
06F9 2104 :
06F9 2105 : R2 -Addr of message buffer
06F9 2106 : R4 -Addr of PDT
06F9 2107 : PDT$L_CNTCDRP(R4) -CDRP holding suspended SYSAP context
06F9 2108 :
06F9 2109 : Outputs:
06F9 2110 :
06F9 2111 : R4 -Preserved
06F9 2112 : Other registers -Destroyed
06F9 2113 : PDT$W_FLAGS(R4) -If PDT$M_CNTRL is set then PDT$M_CNTRL
06F9 2114 : and PDT$M_CNTRBSY are both cleared
06F9 2115 :-
06F9 2116
06F9 2117 .ENABL LSB
06F9 2118
06F9 2119 FPC$REC_RDCNT::
06F9 2120
06F9 2121 BBCC #PDT$V_CNTRL, - ; Branch if no release of
06FB 2122 PDT$W_FLAGS(R4), 10$ ; counters is pending
06FF 2123 BICW #PDT$M_CNTRBSY, - ; Else this is a release --
0701 2124 PDT$W_FLAGS(R4) ; clear counters busy
0704 2125
0704 2126 10$: MOVL PDT$L_CNTCDRP(R4), R5 ; Get SYSAP's CDRP
0709 2127 MOVZWL #SS$_NORMAL, R0 ; Set success status for SYSAP
070C 2128 PUSHL R4 ; Save PDT addr
070E 2129 MOVQ CDRP$L_FR3(R5), R3 ; Get SYSAP's saved R3, R4
0712 2130 JSB @CDRP$L_FPC(R5) ; Call SYSAP back with results
0715 2131 POPL R4 ; Retrieve PDT addr
0718 2132 RSB
0719 2133
0719 2134 .DSABL LSB
```

05 00C0 C4 E5 06F9 2121 BBCC #PDT\$V_CNTRL, - ; Branch if no release of
01 AA 06FB 2122 PDT\$W_FLAGS(R4), 10\$; counters is pending
00C0 C4 06FF 2123 BICW #PDT\$M_CNTRBSY, - ; Else this is a release --
0701 2124 PDT\$W_FLAGS(R4) ; clear counters busy
0704 2125
55 00D4 C4 D0 0704 2126 10\$: MOVL PDT\$L_CNTCDRP(R4), R5 ; Get SYSAP's CDRP
50 01 3C 0709 2127 MOVZWL #SS\$_NORMAL, R0 ; Set success status for SYSAP
54 DD 070C 2128 PUSHL R4 ; Save PDT addr
53 10 A5 7D 070E 2129 MOVQ CDRP\$L_FR3(R5), R3 ; Get SYSAP's saved R3, R4
0C B5 16 0712 2130 JSB @CDRP\$L_FPC(R5) ; Call SYSAP back with results
54 8ED0 0715 2131 POPL R4 ; Retrieve PDT addr
05 0718 2132 RSB
0719 2133
0719 2134 .DSABL LSB

MISC. ROUTINES

```
0719 2136 .SBTTL MISC. ROUTINES
0719 2137 .SBTTL - FPC$CHK_SCONID, CHECK SENDER CONID
0719 2138 .SBTTL - FPC$CHK_DCONID, CHECK DESTINATION CONID
0719 2139 .SBTTL - FPC$CHK_LCONID, CHECK CONID IN LCONID
0719 2140
0719 2141 :+
0719 2142 FPC$CHK_SCONID -- Verifies the sender connection ID in the SCS
0719 2143 header and returns the address of the CDT
0719 2144 FPC$CHK_DCONID -- Verifies the destination connection ID in the SCS
0719 2145 header and returns the address of the CDT
0719 2146 FPC$CHK_LCONID -- Verifies the connection ID in the CONID portion
0719 2147 of an XCT_ID in a block xfer message. (First
0719 2148 longword of XCT_ID)
0719 2149
0719 2150 The connection ID index (l.o. word) is extracted and compared
0719 2151 with the maximum index number. If it exceeds the maximum index,
0719 2152 return error. Else, compute the CDT address from the index.
0719 2153 Check the sequence # in the CDT. If they agree, return success.
0719 2154 Else return error.
0719 2155
0719 2156 Inputs:
0719 2157
0719 2158 R2 -Addr of message/datagram buffer
0719 2159 R4 -Addr of PDT
0719 2160
0719 2161 Outputs:
0719 2162
0719 2163 R0 -1/0 for success/fail
0719 2164 R1 -Destroyed
0719 2165 R2 -Addr of msg/dg (CHK_SCONID)
0719 2166 -Addr of msg/dg iff success (CHK_D/LCONID)
0719 2167 R3 -Addr of CDT if success
0719 2168 Other registers -Preserved
0719 2169 :-
0719 2170
0719 2171 .ENABL LSB
0719 2172
0719 2173 FPC$CHK_SCONID:
0719 2174
0719 2175 MOVZWL SCSSL_SRC CONID(R2),R1 ; Get source connection ID index
0719 2176 MOVL G^SCSSL_GL_CDL,R3 ; Get addr of connx descriptor list
0724 2177 CMPW R1,CDL$W_MAXCONIDX(R3) ; Compare index with maximum
0728 2178 BGTRU BAD_SCONID ; Branch if index is too big
072A 2179 MOVL (R3)[R1],R3 ; Turn index to CDT address
072E 2180 CMPL CDT$W_LCONID(R3),- ; ID in msg/dg matches ID in CDT?
0731 2181 SCSSL_SRC CONID(R2)
0733 2182 BNEQ BAD_SCONID ; Branch if not
0735 2183 MOVZWL #SS$_NORMAL,R0 ; Else success status
0738 2184 RSB
0739 2185
0739 2186 FPC$CHK_LCONID:
0739 2187
0739 2188 MOVL SCSSL_LCONID(R2),R0 ; Extract CONID from message
073D 2189 BRB 10$ ; Join common code
073F 2190
073F 2191 FPC$CHK_DCONID::
073F 2192
```

53 51 FC A2 3C 0719 2175 MOVZWL SCSSL_SRC CONID(R2),R1 ; Get source connection ID index
00000000'GF D0 071D 2176 MOVL G^SCSSL_GL_CDL,R3 ; Get addr of connx descriptor list
FO A3 51 B1 0724 2177 CMPW R1,CDL\$W_MAXCONIDX(R3) ; Compare index with maximum
48 1A 0728 2178 BGTRU BAD_SCONID ; Branch if index is too big
53 6341 D0 072A 2179 MOVL (R3)[R1],R3 ; Turn index to CDT address
18 A3 D1 072E 2180 CMPL CDT\$W_LCONID(R3),- ; ID in msg/dg matches ID in CDT?
FC A2 0731 2181 SCSSL_SRC CONID(R2)
3D 12 0733 2182 BNEQ BAD_SCONID ; Branch if not
50 01 3C 0735 2183 MOVZWL #SS\$_NORMAL,R0 ; Else success status
05 0738 2184 RSB
0739 2185
0739 2186 FPC\$CHK_LCONID:
0739 2187
50 FO A2 D0 0739 2188 MOVL SCSSL_LCONID(R2),R0 ; Extract CONID from message
04 11 073D 2189 BRB 10\$; Join common code
073F 2190
073F 2191 FPC\$CHK_DCONID::
073F 2192


```
- FPC$CHK_LCONID, CHECK CONID IN LCONID

50  F8 A2  D0 073F 2193      MOVL  SCSS$L_DST_CONID(R2),R0  ; Get destination connection ID
      0743 2194
53  51 50 3C 0743 2195 10$: MOVZWL R0,R1  ; Extract index
      00000000 GF D0 0746 2196      MOVL  G^SCSS$GL_CDL,R3  ; Get addr of connx descriptor list
      FO A3 51 B1 074D 2197      CMPW  R1,CDLSW_MAXCONIDX(R3) ; Compare index with maximum
      OE 1A 0751 2198      BGTRU  BAD_CONID ; Branch if index is too big
      53 6341 D0 0753 2199      MOVL  (R3)[R1],R3 ; Turn index to CDT address
      50 18 A3 D1 0757 2200      CMPL  CDT$L_LCONID(R3),R0 ; ID in msg/dg matches ID in CDT?
      04 12 075B 2201      BNEQ  BAD_CONID ; Branch if not
      50 01 3C 075D 2202      MOVZWL #SS$_NORMAL,R0 ; Else success status
      05 0760 2203      RSB ; Return
      0761 2204
      0761 2205 BAD_CONID:
      0761 2206
      F4 A2 B1 0761 2207      CMPW  SCSS$W_MTYPE(R2),- ; Is this an application datagram?
      0B 0764 2208      #SCSS$_APPL_DG
      05 13 0765 2209      BEQL  20$ ; Branch if so
      F896' 30 0767 2210      BSBW  INT$INS_MFREEQ ; Return message buffer to free queue
      03 11 076A 2211      BRB  30$ ; Join common exit
      076C 2212
      F891' 30 076C 2213 20$: BSBW  INT$INS_DFREEQ ; Return dg buffer to free queue
      076F 2214
      50 D4 076F 2215 30$: CLRL  R0 ; Set status to failure
      05 0771 2216      RSB ; Return
      0772 2217
      0772 2218 BAD_SCONID:
      0772 2219
      0772 2220      $DEBUGCHECK #ERR$V_DEB_SCERR ; Optionally, bugcheck on this error
      E8 11 0785 2221      BRB  30$ ; To recover, go return error to caller
      0787 2222
      0787 2223      .DSABL LSB
```



```
0787 2225      .SBTTL FPC$INITIAL, INITIALIZE AT THIS LAYER
0787 2226      .SBTTL - BUILD BDT
0787 2227
0787 2228 ;+
0787 2229 ; The buffer descriptor table is shared among CI ports. If it does
0787 2230 ; not already exist, allocate and initialize it.
0787 2231 ; -
0787 2232
0787 2233 ASSUME CIBDT$L_WAITFL+4 EQ CIBDT$L_WAITBL
0787 2234 ASSUME CIBDT$L_WAITBL+4 EQ CIBDT$L_SIZE
0787 2235 ASSUME CIBDT$L_SIZE+2 EQ CIBDT$L_TYPE
0787 2236 ASSUME CIBDT$L_TYPE+1 EQ CIBDT$L_SUBTYP
0787 2237 ASSUME CIBDT$L_SUBTYP+1 EQ CIBDT$L_FREEBD
0787 2238 ASSUME CIBDT$L_FREEBD+4 EQ CIBDT$L_MAXIDX
0787 2239 ASSUME CIBDT$L_MAXIDX+8 EQ CIBDT$L_BDLIST
0787 2240
0787 2241      .ENABL LSB
0787 2242
0787 2243 FPC$INITIAL::
0787 2244
51 00000000'GF D5 0787 2245      TSTL G^SCS$GL_BDT ; Got buffer descriptors already?
6D 12 078D 2246      BNEQ 40$ ; Branch if so
51 00000000'GF 3C 078F 2247      MOVZWL G^SCS$GW_BDTCNT,R1 ; Get # of buffer descriptors
51 51 04 78 0796 2248      PUSHL R1 ; Save it
51 51 18 C0 079C 2249      ASHL #4,R1,R1 ; Get # bytes of descriptors
00000000'GF 16 079F 2250      ADDL #CIBDT$L_LENGTH,R1 ; + BDT header length
57 50 E9 07A5 2251      JSB G^EXE$ALONONPAGED ; Allocate pool for descriptors
82 52 DD 07A8 2252      BLBC R0,50$ ; Branch if failure
82 FC A2 DE 07AA 2253      PUSHL R2 ; Save addr of BDT
82 82 51 B0 07AD 2254      MOVL R2,(R2)+ ; Set BD wait queue
82 0161 8F B0 07B1 2255      MOVAL -4(R2),(R2)+ ; Listhead empty
62 04 AE D0 07B4 2256      MOVW R1,(R2)+ ; Set structure size,
82 82 D4 07B9 2257      MOVW #<DYN$C_CI_BDTa8 + DYN$C_CI>,(R2)+ ; type, and subtype
62 04 AE D0 07BB 2258      CLRL (R2)+ ; Clear ptr for later
82 D7 07BF 2259      MOVL 4(SP),(R2) ; Set # buffer descriptors
00000000'GF 82 D4 07C1 2260      DECL (R2)+ ; Max index = # BD's-1
51 8ED0 07C3 2261      CLRL (R2)+ ; Clear reserved longwd
50 00000000'GF 51 8ED0 07C3 2262      MOVL R2,G^SCS$GL_BDT ; Save addr in system wide data base
50 50 60 D0 07CD 2263      POPL R1 ; Get BDT address again
50 1A A0 3C 07D4 2264      MOVL G^SCS$GL_CDL,R0 ; Get addr of connx descriptor list
50 5A 8F 8C 07D7 2265      MOVL (R0),R0 ; and addr of first CDT.
50 6E D5 07DB 2266      MOVZWL CDT$L_LCONID+2(R0),R0 ; Get that CDT's sequence number
14 13 07DF 2267      XORB2 #^X5A,R0 ; Make it unique
07E1 2268      TSTL (SP) ; Get # buffer descriptors
07E3 2269      BEQL 30$ ; Branch if zero
07E3 2270
07E3 2271 ;
07E3 2272 ; Loop to initialize buffer descriptors links all BD's onto the
07E3 2273 ; free list rooted at CIBDT$L_FREEBD, marks each BD invalid, and
07E3 2274 ; initializes the BD sequence number
07E3 2275 ;
07E3 2276
02 A2 50 B4 07E3 2277 20$: CLRW CIBD$L_FLAGS(R2) ; Clear valid bit
0C A1 52 D0 07E5 2278      MOVW R0,CIBD$L_KEY(R2) ; Init sequence #
51 52 D0 07E9 2279      MOVL R2,CIBD$L_LINK(R1) ; Link this BD to previous
52 10 A2 DE 07ED 2280      MOVL R2,R1 ; Set this BD to previous
07F0 2281      MOVAL CIBD$L_LENGTH(R2),R2 ; Step to next BD
```


- BUILD BDT

EC 6E	F5	07F4	2282		SOBGTR	(SP),20\$; Branch if more BD's to do
		07F7	2283					
OC A1	D4	07F7	2284	30\$:	CLRL	CIBD\$LINK(R1)		; Zero last fwd link
8E	D5	07FA	2285		TSTL	(SP)+		; Clear stack
50 01	D0	07FC	2286	40\$:	MOVL	#SS\$NORMAL,R0		; Set for succes
	05	07FF	2287	50\$:	RSB			
		0800	2288					
		0800	2289		.DSABL	LSB		
		0800	2290					
		0800	2291		.END			

PAFPCALL
Symbol table

M 5

16-SEP-1984 01:10:45 VAX/VMS Macro V04-00
10-SEP-1984 01:15:44 [DRIVER.SRC]PAFPCALL.MAR;2Page 57
(34)

ALLOC BD	000003BB	R	01
BAD_CONID	00000761	R	01
BAD_SCONID	00000772	R	01
BD_SEQ_ERROR	00000560	R	01
BSY_ERR	000005A3	R	01
BUGS_CIPORT	*****	X	01
CDLSW_MAXCONIDX	= FFFFFFFF0		
CDRPSB_RMOD	= FFFFFFFAB		
CDRPSL_BCNT	= FFFFFFFD2		
CDRPSL_CDT	= 00000024		
CDRPSL_FPC	= 0000000C		
CDRPSL_FR3	= 00000010		
CDRPSL_LBOFF	= 00000030		
CDRPSL_LBUFH_AD	= 0000002C		
CDRPSL_MSG_BUF	= 0000001C		
CDRPSL_RBOFF	= 00000038		
CDRPSL_RBUFH_AD	= 00000034		
CDRPSL_RSPID	= 00000020		
CDRPSL_RWCPT	= 00000028		
CDRPSL_SAVD_RT	= 00000018		
CDRPSL_SWAPTE	= FFFFFFFCC		
CDRPSL_XCT_LEN	= 0000003C		
CDRPSW_BOFF	= FFFFFFFD0		
CDRPSW_STS	= FFFFFFFCA		
CDTSC_RSTATION	= 00000020		
CDTSC_ACCP_PEND	= 00000002		
CDTSC_ACCP_SENT	= 0000000A		
CDTSC_CLOSED	= 00000000		
CDTSC_CON_ACK	= 00000008		
CDTSC_CON_PEND	= 00000001		
CDTSC_CON_REC	= 00000009		
CDTSC_CON_SENT	= 00000007		
CDTSC_CR_PEND	= 00000005		
CDTSC_DCR_PEND	= 00000006		
CDTSC_DISC_ACK	= 00000003		
CDTSC_DISC_MTC	= 00000006		
CDTSC_DISC_PEND	= 00000004		
CDTSC_DISC_REC	= 00000004		
CDTSC_DISC_SENT	= 00000005		
CDTSC_OPEN	= 00000002		
CDTSC_REJ_PEND	= 00000003		
CDTSC_REJ_SENT	= 0000000B		
CDTSL_BYTMAPD	= 00000094		
CDTSL_BYTREQD	= 00000090		
CDTSL_BYTSENT	= 00000088		
CDTSL_CDTLST	= 0000006C		
CDTSL_CRWAITQBL	= 0000003C		
CDTSL_CRWAITQFL	= 00000038		
CDTSL_DGDISCARD	= 00000078		
CDTSL_DGINPUT	= 00000004		
CDTSL_DGRCVD	= 00000074		
CDTSL_DGSENT	= 00000070		
CDTSL_FPC	= 00000064		
CDTSL_FR5	= 00000068		
CDTSL_LCONID	= 00000018		
CDTSL_LPROCNAME	= 00000054		
CDTSL_MSGINPUT	= 00000000		

CDTSL_MSGRCVD	= 00000080		
CDTSL_MSGSENT	= 0000007C		
CDTSL_PB	= 0000001C		
CDTSL_PDT	= 00000010		
CDTSL_RCONID	= 00000014		
CDTSL_REQDATS	= 0000008C		
CDTSL_RPROCNAME	= 00000050		
CDTSL_SCSMSG	= 0000002C		
CDTSL_SNDAT	= 00000084		
CDTSL_DGRC	= 0000004C		
CDTSL_INITLREC	= 00000048		
CDTSL_MINREC	= 00000044		
CDTSL_PENDREC	= 00000046		
CDTSL_QBDT_CNT	= 0000009A		
CDTSL_QCR_CNT	= 00000098		
CDTSL_REASON	= 00000026		
CDTSL_REC	= 00000042		
CDTSL_SEND	= 00000040		
CDTSL_STATE	= 00000028		
CHK_CRWAIT	= 000006CB	R	01
CIBDSC_LENGTH	= 00000010		
CIBDSL_BLEN	= 00000004		
CIBDSL_CDRP	= 0000000C		
CIBDSL_LINK	= 0000000C		
CIBDSL_SWAPTE	= 00000008		
CIBDSM_AC	= 00001000		
CIBDSM_V	= 00008000		
CIBDSV_ACMOD	= 0000000D		
CIBDSV_V	= 0000000F		
CIBDSW_FLAGS	= 00000000		
CIBDSW_KEY	= 00000002		
CIBDTSC_SUBTYP	= FFFFFFFF3		
CIBDTSC_TYPE	= FFFFFFFF2		
CIBDTSC_BDLIST	= 00000000		
CIBDTSC_LENGTH	= 00000018		
CIBDTSL_FREEBD	= FFFFFFFF4		
CIBDTSL_MAXIDX	= FFFFFFFF8		
CIBDTSL_WAITBL	= FFFFFFFEC		
CIBDTSL_WAITFL	= FFFFFFFE8		
CIBDTSW_SIZE	= FFFFFFFF0		
CIBHANSI_BNAME	= 00000004		
CIBHANSI_BOFF	= 00000000		
CIBHANSI_RCONID	= 00000008		
CNFS_LKP_PB_MSG	*****	X	01
CNFS_STOP_VCS	*****	X	01
COMMON_XFER	= 000004DF	R	01
CON_MEM_FAIL	= 00000047	R	01
CON_MEM_FAIL1	= 00000049	R	01
DGCOM	= 00000624	R	01
DG_ALC_FAIL	= 000002E2	R	01
DISC_CON_ACK	= 0000011C	R	01
DISC_DISC_REC	= 0000013B	R	01
DISC_ILLSTATE	= 0000010D	R	01
DISC_OPEN	= 0000014F	R	01
DQUEUE_DG	= 0000031A	R	01
DQ_INCOMPLETE	= 00000333	R	01
DYNISC_CI	= 00000061		

PAFPCALL
Symbol table

N 5

16-SEP-1984 01:10:45 VAX/VMS Macro V04-00
10-SEP-1984 01:15:44 [DRIVER.SRC]PAFPCALL.MAR;2

Page 58
(34)

```

DYN$C CI BDT          = 00000001
ERR$BUGCHECK          ***** X 01
ERR$BUGCHECKNF        ***** X 01
ERR$CRASHVC           ***** X 01
ERR$DEBUGCHECK        ***** X 01
ERR$DISC_PWFAIL       ***** X 01
ERR$V_DEB_SCERR       ***** X 01
ERR$V_DEB_XCTER       ***** X 01
EXESA$CONONPAGED      ***** X
FPC$ACCEPT            00000053 RG 01
FPC$ALLOCDG           000002D4 RG 01
FPC$ALLOCMMSG         0000015F RG 01
FPC$CHK_DCONID        0000073F RG 01
FPC$CHK_LCONID        00000739 R  01
FPC$CHK_SCONID        00000719 R  01
FPC$CONNECT           00000006 RG 01
FPC$DCONNECT          000000CF RG 01
FPC$DEALLOCDG         000002E8 RG 01
FPC$DEALLOMSG         00000218 RG 01
FPC$DEALRGMMSG        0000022B RG 01
FPC$INITIAL           00000787 RG 01
FPC$MAINTFCN          00000000 RG 01
FPC$MAP               000003AE RG 01
FPC$MAPBYPASS         00000395 RG 01
FPC$MAPIRP            0000039E RG 01
FPC$MAPIRPBYP         0000038D RG 01
FPC$MRESET            000005C4 RG 01
FPC$MSTART            000005CC RG 01
FPC$QUEUEDG           000002EE RG 01
FPC$QUEUEMDGS         000002F6 RG 01
FPC$RCHMSGBUF         000001C3 RG 01
FPC$RCLMSGBUF         000001CD RG 01
FPC$READCOUNT        0000057A RG 01
FPC$REC_CNFRREC       00000638 RG 01
FPC$REC_DATREC        00000638 RG 01
FPC$REC_DGREC         000005FF RG 01
FPC$REC_MSREC         00000699 RG 01
FPC$REC_RDCNT         000006F9 RG 01
FPC$REC_SNDG          0000061B RG 01
FPC$REC_SNDMSG        000006E6 RG 01
FPC$REJECT            000000B6 RG 01
FPC$REQDATA           00000453 RG 01
FPC$RLSCOUNT          000005BB RG 01
FPC$SENDATA           0000049A RG 01
FPC$SENDG             0000034C RG 01
FPC$SENDMSG           00000266 RG 01
FPC$SENDRGD           0000033F RG 01
FPC$SNDCNTMSG         0000026D RG 01
FPC$STOP_VCS          000005FB RG 01
FPC$UNMAP             000005CC RG 01
FPC_SUCCESS           000002DE R  01
INT$ALLOC_DG          ***** X 01
INT$ALLOC_MSG         ***** X 01
INT$CRASH_PORT        ***** X 01
INT$DEAL_DG           ***** X 01
INT$DEAL_MSG          ***** X 01
INT$DFQ2POOL          ***** X 01

```

```

INT$INS_DFREQ         ***** X 01
INT$INS_DFREQX        ***** X 01
INT$INS_MFREQ         ***** X 01
INT$MRESET            ***** X 01
INT$MSTART            ***** X 01
INT$READCNT           ***** X 01
INT$REQDAT            ***** X 01
INT$SNDG              ***** X 01
INT$SNDG              ***** X 01
INT$SNDMSG            ***** X 01
IRPSM_PAGIO           = 00000004
IRPSM_SWAPIO          = 00000040
ISSUE_RDCNT           0000058F R 01
MAP_COMMON            000003B7 R 01
MEM_ERR               000005E5 R 01
PBSC_PWR_FAIL         = 00004000
PBSC_VC_FAIL          = 00008000
PB$C_CDTLST           = 00000034
PB$W_STATE            = 00000012
PDT$C_CNTCDRP         = 000000D4
PDT$C_MSGHDRSZ        = 000000B4
PDT$C_WAITQBL         = 000000B0
PDT$C_CNTBSY          = 00000001
PDT$C_CNTRL           = 00000002
PDT$C_CNTOwner        = 000000C4
PDT$V_CNTBSY          = 00000000
PDT$V_CNTRL           = 00000001
PDT$W_FLAGS           = 000000C0
PORT_ERR              000005F2 R 01
QUEUE_DG              000002FE R 01
Q_INCOMPLETE          00000336 R 01
Q_SUCCESS              00000316 R 01
RD$C_CDRP             = 00000000
RD$W_SEQNUM           = 00000006
RD_SEQ_ERR            0000067A R 01
SC$SALL_ALLBUF        ***** X 01
SC$SALL_ALLBUF2       ***** X 01
SC$SCOPY_ACCP         ***** X 01
SC$SC_APPL_DG         = 0000000B
SC$SC_APPL_MSG        = 0000000A
SC$SC_OVHD            = 0000000E
SC$SDEALL_CDT         ***** X 01
SC$SDEALL_RSPID       ***** X 01
SC$SDEAL_ALLBUF       ***** X 01
SC$SDEAL_SCSREC       ***** X 01
SC$SDISC_VCFAIL       ***** X 01
SC$SGL_BDT            ***** X 01
SC$SGL_CDL            ***** X 01
SC$SGL_RDT            ***** X 01
SC$SGW_BDTCNT         ***** X 01
SC$SGW_FLOWCUSH       ***** X 01
SC$SGW_MAXMSG         ***** X 01
SC$SL_DST_CONID       = FFFFFFFF8
SC$SL_LCONID          = FFFFFFFF0
SC$SL_REC_BOFF        = 00000008
SC$SL_REC_NAME        = 00000004
SC$SL_RSPID           = FFFFFFFF4

```


PAFPCALL
Symbol table

```
SCSSL_SND_BOFF      = 00000000
SCSSL_SND_NAME      = FFFFFFFFC
SCSSL_SRC_CONID     = FFFFFFFFC
SCSSL_XCT_LEN       = FFFFFFFF8
SCSSMAP_VMSSTS      ***** X 01
SCSSREC_SCSSMSG     ***** X 01
SCSSREQ_SCSSSEND    ***** X 01
SCSSRESOMEWAITR     ***** X 01
SCSST_DST_PROC      = 00000004
SCSST_SRC_PROC      = 00000014
SCSSW_CREDIT        = FFFFFFFF6
SCSSW_LENGTH        = FFFFFFFF0
SCSSW_MTYPE         = FFFFFFFF4
SCSSEND             = 000000B0 R 01
SC_SEQ_ERR          = 000006EE R 01
SS$_ABORT           = 0000002C
SS$_DGQINCOMP       = 000009C0
SS$_ILLCDTST        = 00002154
SS$_ILLIOFUNC       = 000000F4
SS$_INSMEM          = 00000124
SS$_INTERLOCK       = 0000038C
SS$_NORMAL          = 00000001
STATE_CDT           = 00000698 R 01
STATE_ERR           = 00000574 R 01
STATE_ERR_R3        = 00000571 R 01
SUSP_CONCALL        = 00000568 R 01
SYSAP$C_DGREC       = 00000000
SYSAP$C_DGSNT       = 00000001
SYSAP$C_DISPRET     = 00000001
WAIT_BD             = 0000040E R 01
```

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$\$\$115_DRIVER	00000800 (2048.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
\$AB\$\$	00000000 (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.04	00:00:00.97
Command processing	109	00:00:00.45	00:00:02.57
Pass 1	431	00:00:11.37	00:00:39.58
Symbol table sort	0	00:00:01.40	00:00:04.26
Pass 2	389	00:00:03.68	00:00:16.82
Symbol table output	10	00:00:00.15	00:00:00.31
Psect synopsis output	0	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	971	00:00:17.10	00:01:04.52

The working set limit was 2100 pages.
99428 bytes (195 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1336 non-local and 70 local symbols.
2291 source lines were read in Pass 1, producing 23 object records in Pass 2.
39 pages of virtual memory were used to define 37 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-----	-----
\$255\$DUA28:[DRIVER.OBJ]PALIB.MLB;1	7
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	16
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	29

1486 GETS were required to define 29 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LISS:PAFPCALL/OBJ=OBJ\$:PAFPCALL MSRC\$:PAFPCALL/UPDATE=(ENH\$:PAFPCALL)+EXECML\$/LIB+LIB\$:PALIB.MLB/LIB

0114 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PAINT
LIS

PAFPCALL
LIS

PAINT
LIS

PAMONIT
LIS